ADS Synthesis User Manual

(Version 2.1)

Document version revision record

VERSION NUMBER	RELEASE DATE	REVISION HISTORY
V1.0	2022.07.12	initial version
V2.0	2022.09.21	Add the ADS Integrated Compile configuration option "Multiple File Compilation Unit" description
V2.1	2022.10.25	Update NV related description
V2.3	2022.12.01	Add full_case, parallel_case, translate_off_on , syn_state_machine related description

Directory

Figure catalog

1. ADS FLOW OVERVIEW

ADS (Architecture-Driven Synthesis) is a tool for logical synthesis of design source files. ADS consists of two main flows, Compile and Synthesize. The compile phase generates the source code RTL netlist, the synthesize phase generates the technology netlist based on the DB of the compile, mapping and optimization. After the compilation phase, the technology netlist is generated.

After completing the ADS flow, the synthesize tool outputs the technology netlist file (.vm), etc. The generated vm The generated vm net list file can be used to perform validation tasks with third-party validation tools. The net list information is passed through the DB file, and the logical synthesis result is also passed to DeviceMap through the generated DB file. The result of logical synthesis is also the net list information passed to DeviceMap through the generated DB file.

The following figure shows the implementation flow of the ADS Synthesize tool, which currently supports Verilog only.

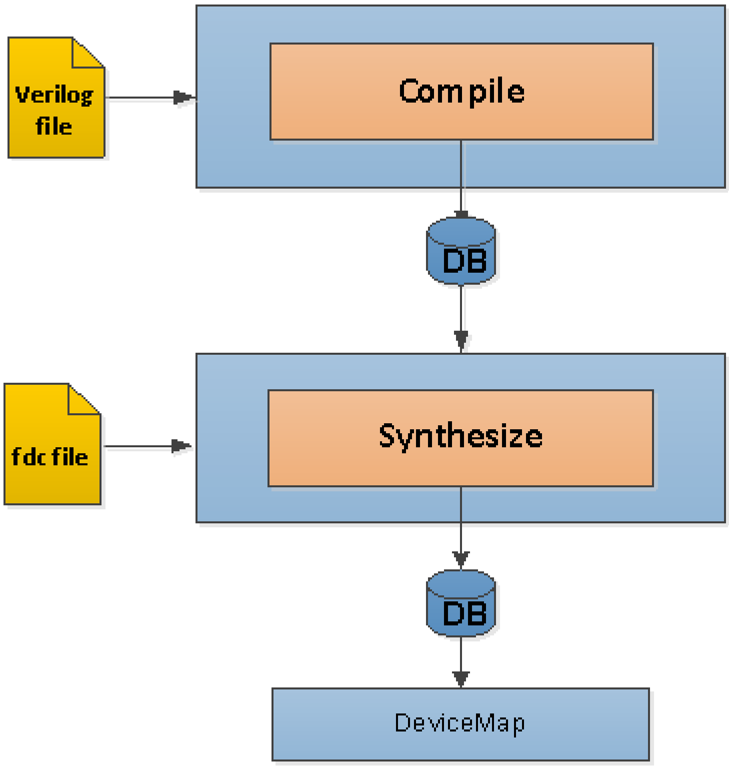


Figure 1-1 ADS Synthesis Design Flow

After the ADS synthesis is completed, the run.log process file is created in the project directory to store all log information. Two subdirectories are created in the project file directory, compile and synthesize.

The main file is <top_module>_rtl.adf, which can be used to display the RTL schematic and is also the DB file for the synthesize phase.

The synthesize directory contains the results of the synthesize phase, and the main files are <top_module>_syn.adf, <top_module>_syn.vm, <top_module>.snr, and so on. The <top_module>_syn.adf file can be used to display the technology schematic. vm file is the map followed by the The snr file is the combined resource and timing report. The contents of this file are included in the run.log file.

The formal.pvf file is generated during both the compile and synthesize phases and is used to guide the formal verification.

2. BASIC OPERATION OF ADS SYNTHESIS

2.1 The ADS synthesis process

After the project file is created, enter the main interface, if the synthesis tool is not ADS, you can switch to ADS synthesis tool. In the project management area as shown in Figure 2-1, right click on the red area and select Project Settings, the Project Settings window will pop up. Select ADS for Synthesize tool and click OK to finish the setting as shown in Figure 2-2.

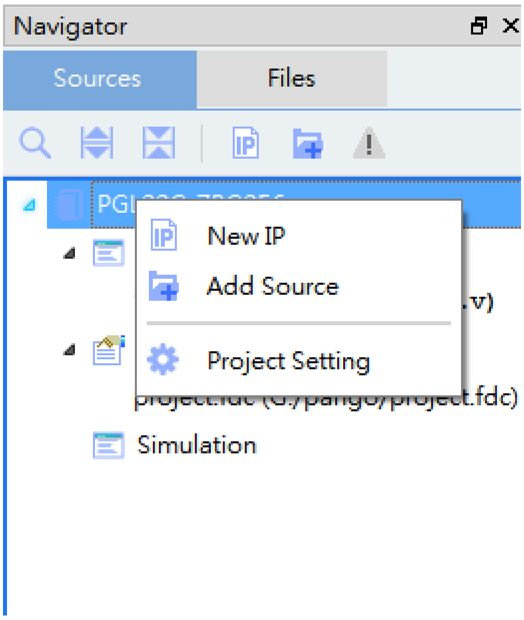


Figure 2-1 ADS Integrated Tool Setup Menu

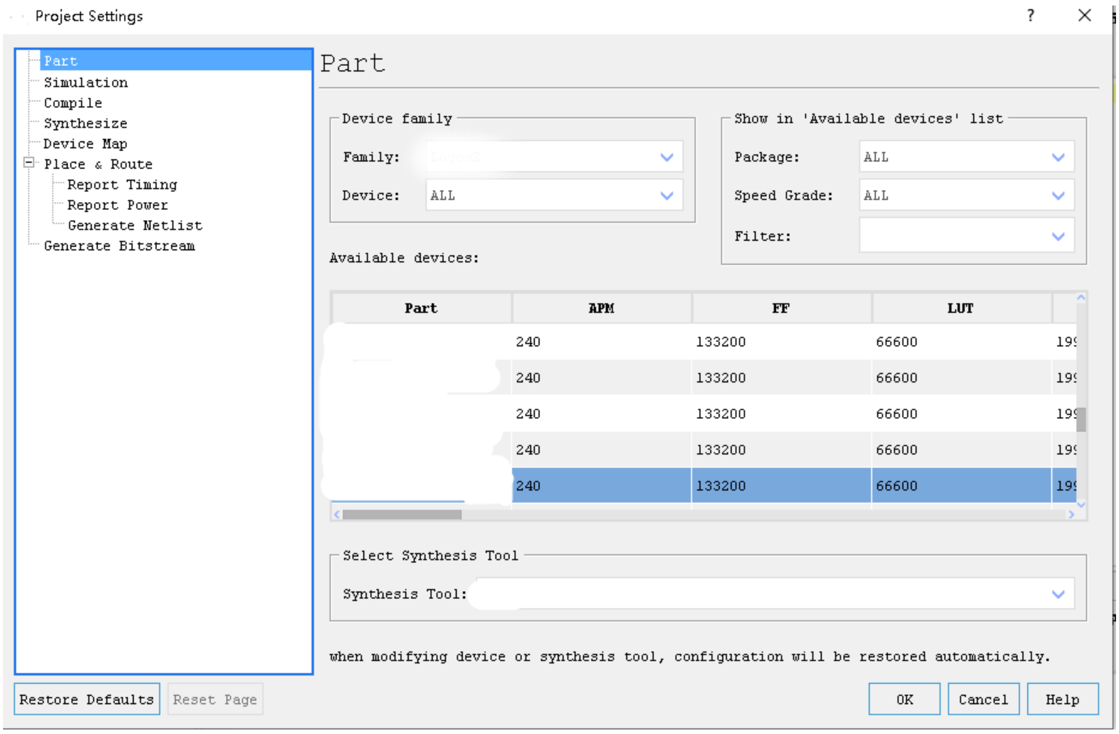


Figure 2-2 ADS Integrated Tool Setup Window

Once the Project Setting is set to ADS, the ADS Flow is displayed as shown in Figure 2-3. The ADS flow consists of two parts, Compile and Synthesize, and the two actions can be executed separately. If only Compile is to be executed, right-click the Compile in the Flow view and click Run to execute it. The RTL netlist will be generated after execution.

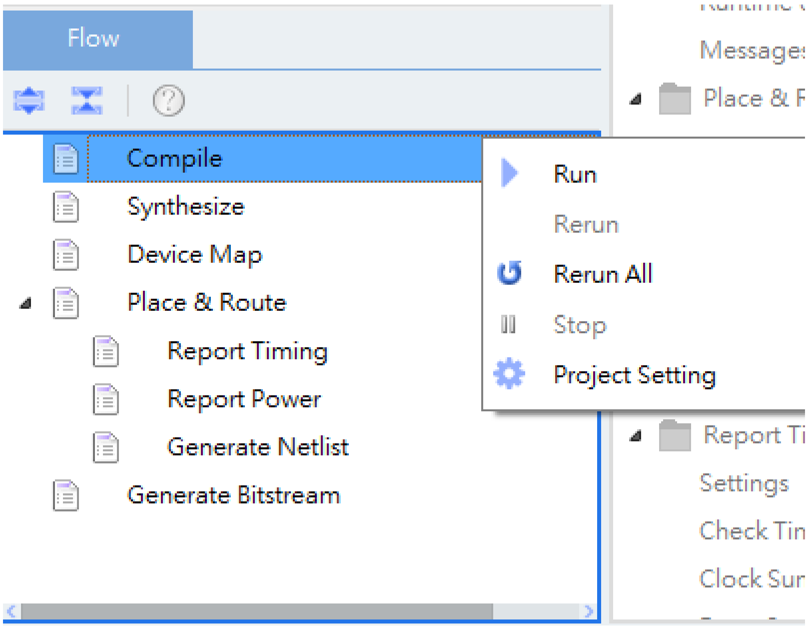


Figure 2-3 ADS Flow

If Compile and Synthesize are successful, as shown in Figure 2-4, ✔ appears on the left side of the corresponding action and ✖ indicates an error.

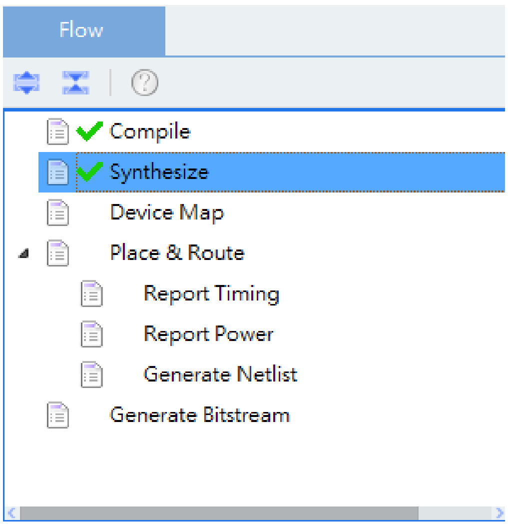


Figure 2-4 Flow Interface after ADS Synthesis is Complete

Figure 2-5 shows that the ADS Synthesis Report link can be opened by clicking on it, and the following screen can be seen by clicking on Synthesize/Settings.

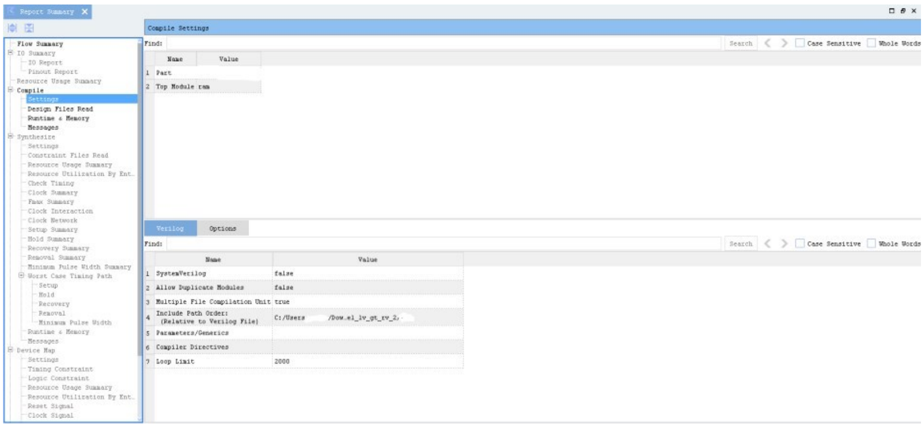


Figure 2-5 ADS Report Summary

In the synthesis report directory there is a file with the extension *.ccr which contains all report information related to the constraints generated during the check fdc constraint synthesis.

2.2 Adding ADS Source Files

1. The design files supported by the ADS synthesis process include a mixture of pure Verilog files, Verilog files, and ADS synthesized intermediate files (*_syn.adf).

2. Constraints supported by the ADS synthesis process include fdc, scf, and lcf files. fdc is used in the synthesize phase, fdc is used in the synthesize phase, and scf and lcf are used in the dev_map phase.

The added files are shown in Figure 2-6 below.

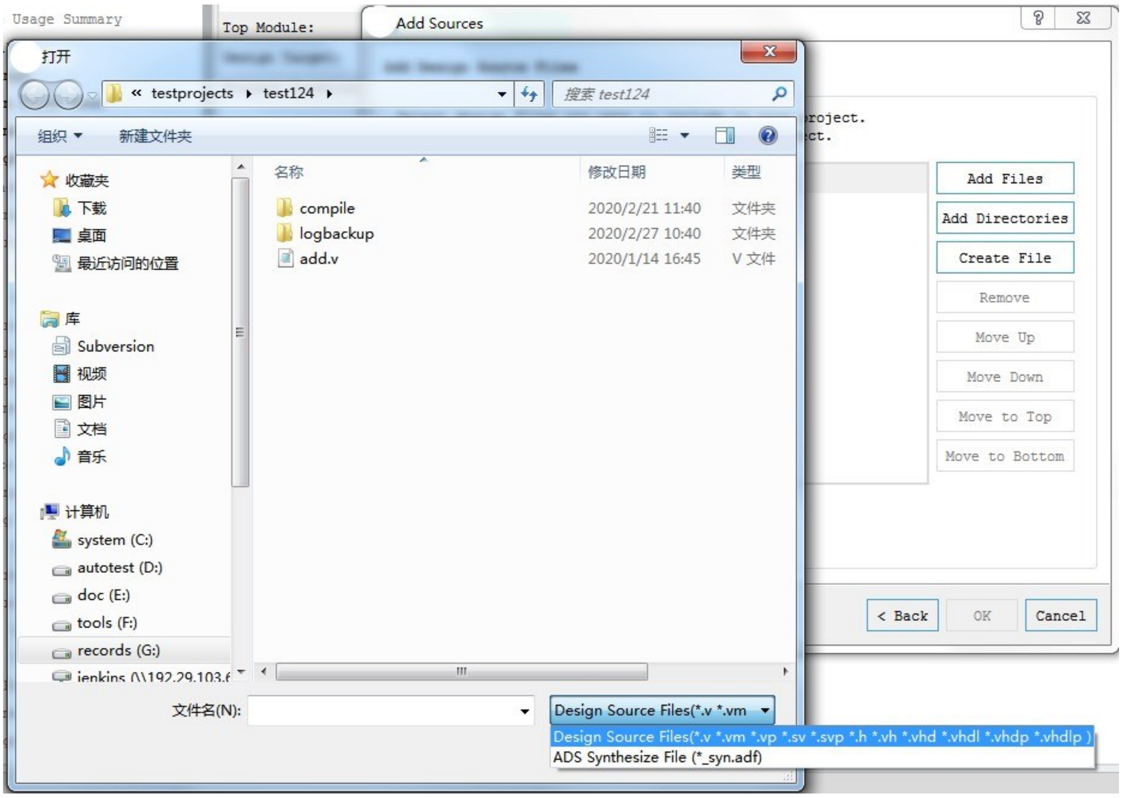


Figure 2-6 ADS Adding Source Files

2.2.1 Pure Verilog Source File

The ADS currently supports Verilog files as source files, which can be added using the Add Source button shown in Figure 2-1.

2.2.2 Mixed Verilog and ADF Source Files

The ADS process also supports Verilog files and ADS synthesized intermediate files (*_syn.adf files) together as source files. When using this approach as a design in a project, the verilog file is used as the top level The ADS synthesized intermediate file (*_syn.adf file) is used as a sublayer that is synthesized from the original project's The submodule is built in the same way as the .v and vm mixins, using /*sync in the top level. The submodule is built in the same way as the .v and vm mixins, which requires the submodule to be marked with the /*synthesis syn_black_box */ attribute in the top layer, as shown in the following figure:

```
module pgr_apb_mif_16bit(
    input                   clk,
    input                   rst_n,

    input          [23:0]   addr,
    input          [15:0]   data,
    input                   we,
    input                   cmd_en,
    output                  cmd_done,

    output    reg    [7:0]  fifo_data,
    input                   fifo_data_valid,
    output    reg              fifo_data_req,

    output   reg [23:0]   p_addr,
    output   reg [15:0]   p_wdata,
    output   reg          p_ce,      //p_sel
    output   reg          p_enable,
    output   reg          p_we,
    input                 p_rdy,
    input        [15:0]   p_rdata
) /* synthesis syn_black_box */ ;
endmodule
```

Figure 2-7 Black Box Module

The *_syn.adf file is used as a submodule, so when you synthesize the adf, you must check the Disable I/O insertion option in the Synthesis Configuration. As shown in the following figure :

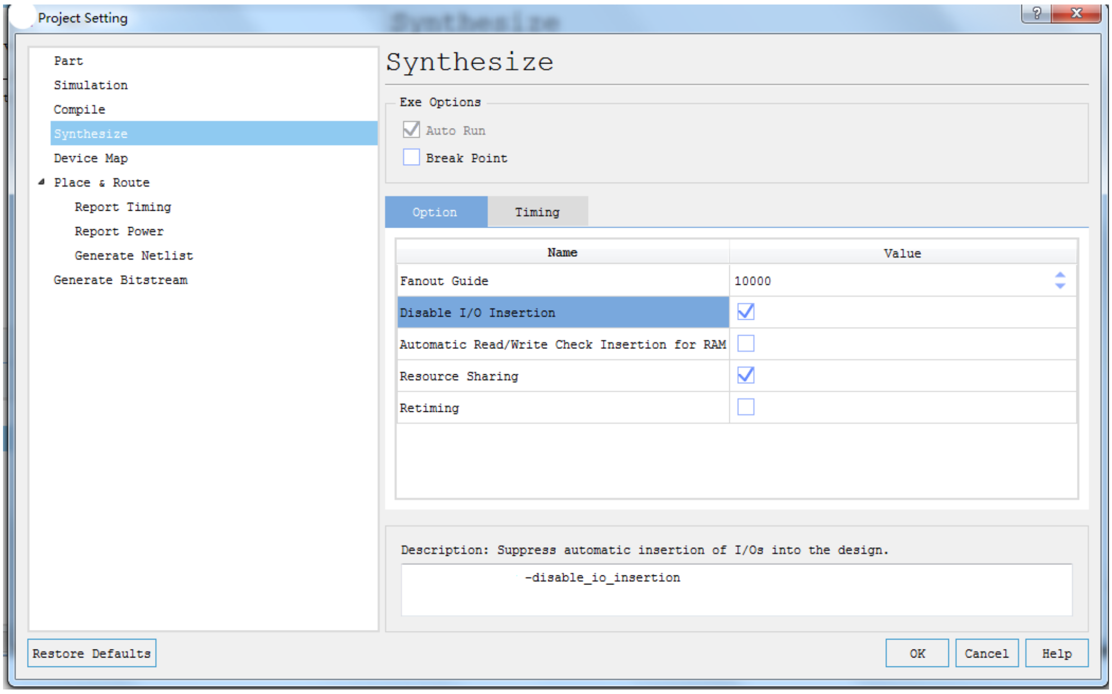


Figure 2-8 Disabling I/O Insertion

Example.

Existing top layer top, submodule A,B. First, use ADS (you must check the Disable I/O Insertion option in the Synthesize configuration option, see Figure 2-8) to synthesize the A,B module, generating A_syn.adf, B_syn.adf.

Then add the A,B module declaration at the top level (you need to add /*synthesis syn_black_box */ description, see Figure 2-7).

Then add the A_syn.adf and B_syn.adf files to the design source.

Finally, you can start synthesizing the top layer (note that the disable I/O insertion option is unchecked in the synthesis configuration). At this point, the sub-layer adf file is not synthesized as a black box, and the top-layer synthesis result is combined with the sub-layer adf file during device mapping. The top layer synthesis result is merged with the sub-layer adf file in the device map and passed to the layout routing.

Since the sublayer adf is used as a black box during synthesis, the object information in the sublayer adf is not available at this stage. Therefore, it is not possible to constrain the object in the sublayer adf in the fdc file. When the sub-layer adf is loaded during the dev_map phase, the timing constraint information in the sub-layer adf is cleared (the top module is not affected); therefore, the scf file is introduced to constrain the object in the sub-layer adf. Therefore, the scf file is introduced to constrain the timing of the sub-layer adf; similarly, the lcf file is introduced to constrain the position and attributes of the sub-layer adf. When writing constraints in the scf and lcf files, the constraint rules are the same as in the fdc file. Note that the hierarchy of the object is the same as the hierarchy of the sub-layer adf before it is integrated. Note that the hierarchy of the object is the same as the hierarchy of the adf sublayer before synthesis. It is not currently supported to constrain the objects in the adf sub-layer directly in UCE and save them in scf and lcf files. The constraints in scf and lcf files are also not recognized when UCE is opened. If you want to constrain the object in scf and lcf files, only manual constraints can be written in the files.

2.3 Project Settings

Compile_settings Click the Project Settings option on the right-click menu of any flow step in the flow display area to open the Project Settings window shown in the figure below. The Settings window in the figure below allows you to set the Compile and Synthesize options.

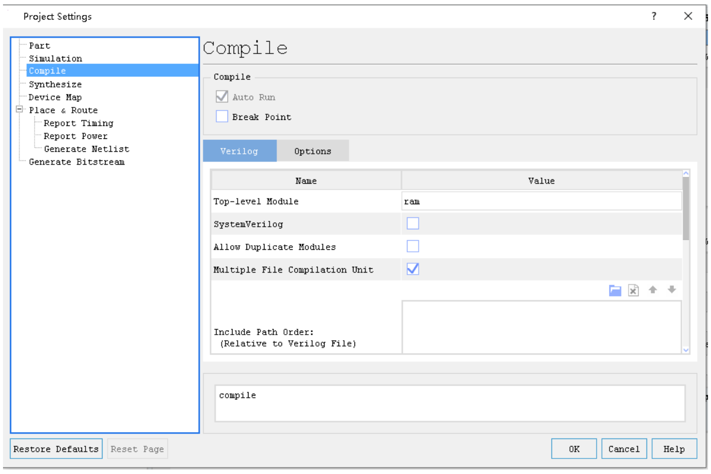


Figure 2-9 Compile Settings Window

Check the Compile in Browser box on the left and set the Compile option on the right.

System Verilog: Unchecked by default. If checked, the default Verilog is set to System Verilog, which must be checked for the design to compile successfully if the syntax of System Verilog is present. This is equivalent to the -system_verilog function in the shell command.

Allow duplicate modules: Unchecked by default. If unchecked, an error is reported if a module with the same name exists in the design. If this option is selected, the module with the same name can be successfully compiled and the software will overwrite the previous module with the latter module in the order of module resolution.

Note: For multiple IP instantiation scenarios, the software will not check the renamed module even if it is not checked and will overwrite the previous module with the later module in module parsing order.

Multiple file compilation unit: Controls whether the variables and macros defined in the current RTL file can take Function in other RTL files. If unchecked, each file will be compiled into a separate compilation unit so that macros will only take Function in the current file. It is equivalent to the multi_file_compilation_unit function of the shell command.

Top level module: Set the top-level module, value is the name of the module. If not set, PDS will automatically select the uninstantiated module as the top module. If there are multiple uninstantiated modules in the input file, ADS will select the first uninstantiated module in alphabetical order as the top module.

Include path order: Set the include path of the project, which is equivalent to the -include_path function of the shell command. Click the button to bring up the Select Directory prompt box, select the include path to finish setting the include path. This option will only include the file directory, not the files under the directory, the actual operation of include file must be combined with the `include statement in the rtl file.

An include file can take several forms:

- include absolute paths, such as adding the `include "/user/dirA/fileA.v" statement directly to the rtl file.
- Include relative paths, such as `include "... /fileA.v". The relative path is based on the directory of the rtl file where the include statement is located, and you must ensure that the relative path between the rtl file where the statement is written and fileA.v is correct.
- With the path of the compile configuration option, such as adding the `include "fileA.v" statement to the rtl file and adding the directory where fileA.v is located to the Include Path Order configuration option.

Note: The include file is only a macro definition file, not a Verilog file, so the include file does not need to be added to the project.

Parameters/Generics : Default is empty. Generic Name is the top-level parameter name, Override Value is the new parameter value, click "+" to add, use space to separate parameter name and parameter value, and use quotes for string type parameter values. For example, {ADDR_WIDTH 10 ASYNC "TRUE"};. Equivalent to the -defined_parameters function of the shell command. extract Top Parameters extracts the top-level module parameter to the GUI interface. If there is no parameter in the top-level module, it will prompt that there is no parameter to extract.

Compiler directives : Default is empty. Used to set macro definition parameters, use '=' between macro definition name and macro definition

Use '=' between macro definition name and macro definition value, and space between multiple macro definition directives, e.g. "ADDR_WIDTH=8 DATA_WIDTH=4 ASYNC". Equivalent to the -compiler_directives function of the shell command.

Loop Limit: Set the upper limit for the number of loops of the for statement, the default value is 2000, which is equivalent to -loop_limit of the shell command.

FSM Compiler: switch to select the FSM encoding format. The default is auto, other optional values are listed in the following table, equivalent to the -fsm_compiler function of the shell command.

ENCODING	DESCRIPTION
auto	Select the optimal encoding method according to the software optimization strategy, the default is one_hot
off	Does not execute FSM infer
one_hot	Only one bit of each state is 1 , which can reduce combinational logic, is the fastest, can optimize timing, but uses more registers
gray	Only one state flips when adjacent states transition, and one_hot uses fewer registers compared to
sequential	Natural number encoding, using the least number of registers under the same conditions
original	Keep the original encoding in the source file
safe	Prevent the state machine from being locked in an illegal state, the default is auto, safe encoding. Once the FSM enters any illegal state, you can pass the safe logic The first time will FSM Jump back to the legal state, which can enhance the stability of the design
safe , one_hot	Use one-hot encoding and prevent the state machine from being locked in an illegal state
safe , gray	Use Gray code encoding and prevent the state machine from being locked in an illegal state
safe , sequential	Use natural digital encoding and prevent the state machine from being locked in an illegal state
safe , original	Keep the original coding in the source file and prevent the state machine from being locked in an illegal state

Synthesize_settings In Figure 2-9, select Synthesize in the Browser on the left, and set the option for Synthesize on the right. Synthesize supports seven settings: Fanout Guide, Disable I/O Insertion, Enable Advanced LUT Combining, Automatic Read/Write Check Insertion for RAM, Resource Sharing, Retiming, and Minimum Control Set Size.

Fanout Guide: Set the maximum global fanout value, the default value is 10000, which is equivalent to the -fanout_guide function in the shell command.

Disable I/O insertion: Switch for disabling I/O buffer insertion; by default, the switch is off and I/O buffers are inserted in the net list; checking this checkbox disables the insertion of I/O buffers in the net list.

Enable Advanced LUT Combining: Switch to combine two compatible LUTs into one dual output GTP_LUT6D. Only available for series with GTP_LUT6D function, i.e. currently only Logos2/Titan2/Titan3 series are supported. The default is on and the default is grayed out outside the Logos2/Titan2/Titan3 series. Equivalent to the -enable_prepacking function of the shell command.

Automatic read/write check insertion for RAM: Corresponds to the -rw_check_on_ram function of the shell command.

Description: Currently, only simple dual-port RAM triggers transparent_write mode.When this option is checked, bypass logic is inserted to ensure that the data read is the data written in the same clock cycle to avoid inconsistent emulation.

Resource sharing: Resource sharing switch. The default is on and it enables resource sharing for

arithmetic operations, mainly for addition (subtraction) and multiplication operations. Resource sharing optimizes area, but can degrade timing. Set to unchecked to disable resource sharing. Equivalent to the -resource_sharing function of the shell command.

Pipelining: Pipelining switch. The default is on, which reduces resources and optimizes area by moving registers forward and absorbing the mux chain on the dpram asynchronous output port. If set to off, no pipelining is performed. Equivalent to the -pipelining function of the shell command.

Retiming: Retiming switch; off by default; when turned on, retiming optimizes the timing, power, or area of the circuit by moving existing registers in the circuit. No additional logic is inserted, but the number of registers may be increased or decreased as the registers are moved. Equivalent to the -retiming function of the shell command.

Minimum control set size: Sets the minimum value of FFs with the same control port, i.e., the minimum value of FFs with consistent ports such as CLK, CE, and synchronous SET/RESET. The default value is 2 and the optional value is a positive integer. If the FF with the same control port is greater than or equal to the set value, no optimization is performed; if it is less than the minimum value, some or all of the control pins of the corresponding FF are shifted to the D input. This function is equivalent to the -min_controlset_size shell command.

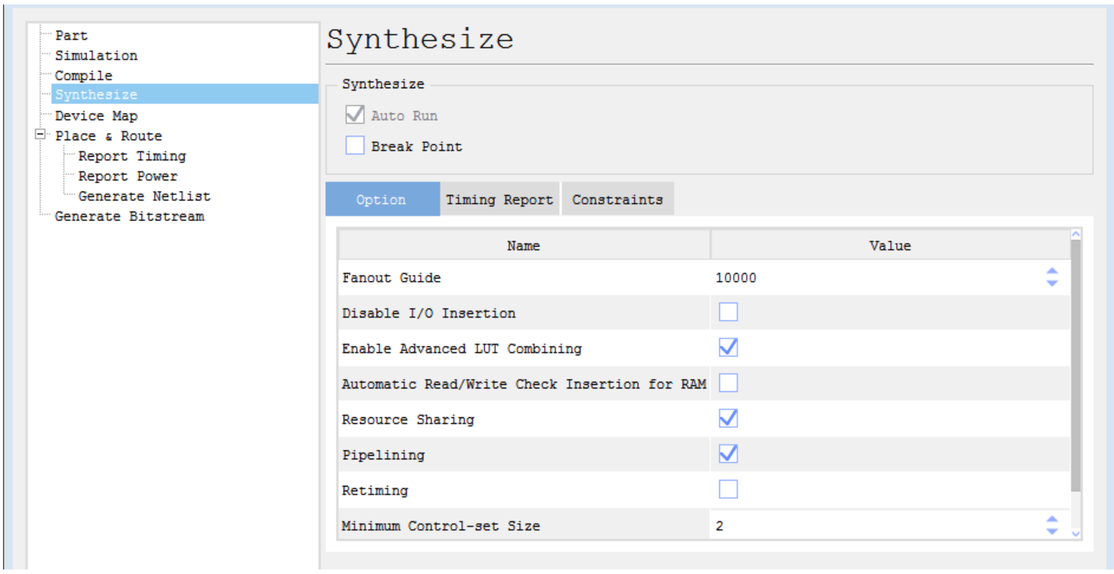


Figure 2-10 Synthesize Options Configuration

The Synthesize timing report supports the following options.

Total Number of Paths (-max_path): i.e., max_path specifies the number of worst-case timing paths reported in the Synthesize Timing Report. This is equivalent to the -max_path function of the shell command.

Number of Paths per Endpoint (-nworst): nworst specifies the maximum number of paths per endpoint that can be stored in the integrated timing analysis. This is equivalent to the -nworst function of the shell command.

Report Paths with Logic Levels >: specifies that the logic levels of each setup/recovery path reported during timing synthesis must be greater than the specified value, and that paths below this value are not displayed. This setting does not affect hold/remove analysis timing paths. If Logic Level > -1 is set, the timing analysis will report the timing paths with the higher logic level first, not necessarily the paths with the worse slack, and the timing report will be sorted by the size of the logic level. (available in TA only)

Limit Paths by Clock Group: Specify the clock group name, and only the paths of the specified clock group are displayed in the timing report. (only available in TA)

Number of Carry Instances as Logic Level: Set the number of carry instances as one logic level, the default is 4 carry instances as one logic level.(available in Post Synthesize TA only)

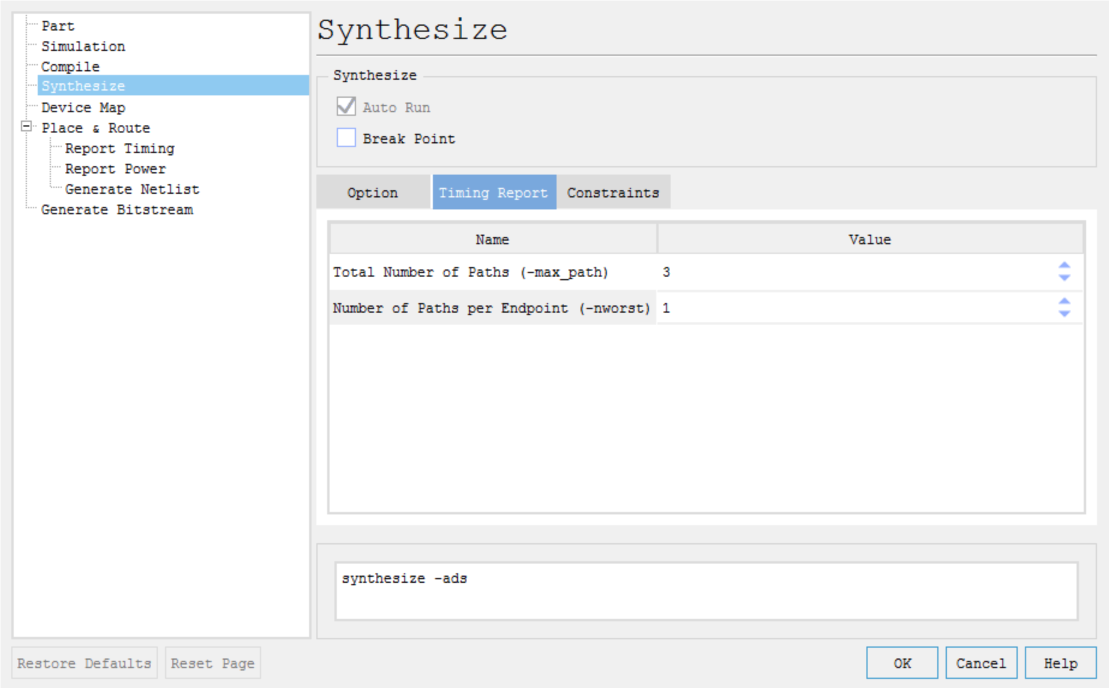


Figure 2-11 Configuring Synthesize Timing Report Options

Select Constraints on the left side of the Synthesize page to set options related to the synthesized timing constraints.

There are two options supported by Constraints.

Frequency (MHz): Used in infer clock, if the correct clock frequency cannot be calculated based on the PLL parameters and the reference clock, this option is used to generate a clock with the specified clock frequency value, default 1.0000 is checked. Auto Constrain is currently not supported, Auto Constrain is grayed out, and Auto Constrain is unchecked. Corresponds to the -frequency function of the shell command.

Use clock period for unconstrained IO: This option adds IO delay constraints to ports where the user does not have IO delay constraints, improving timing constraint coverage. This is equivalent to the -use_clocked_period_for_unconstraint_io function of the shell command.

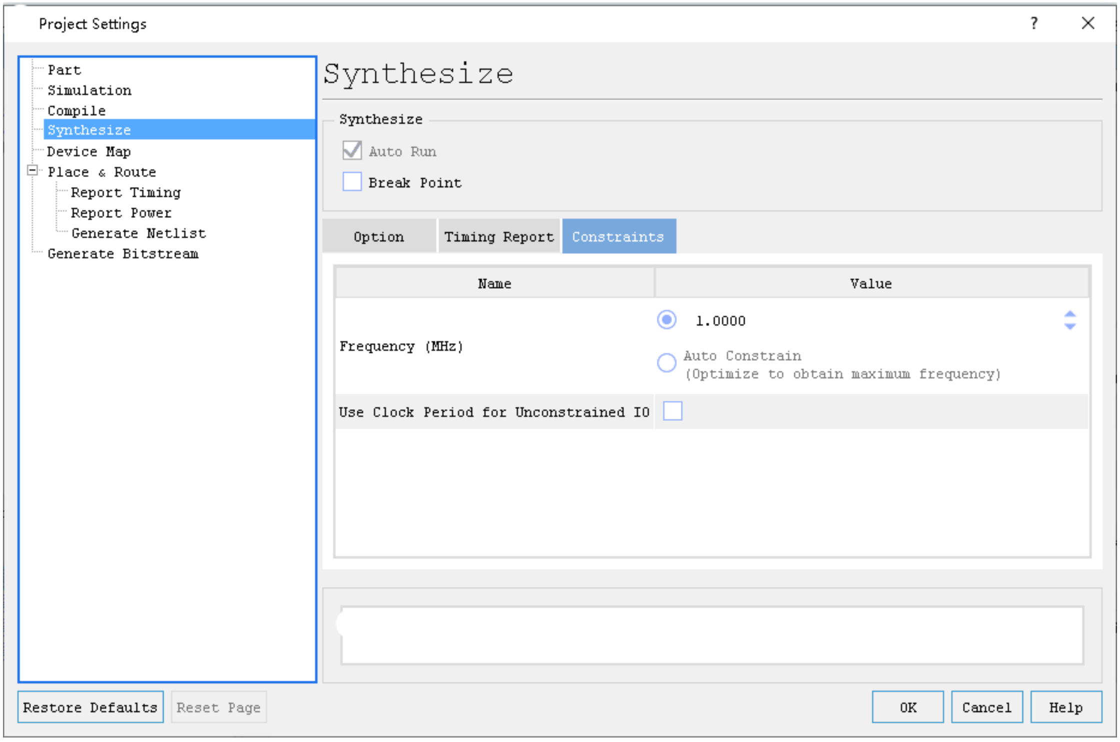


Figure 2-12 Synthesize Constraints option configuration

3. ADS INTEGRATED NETLIST ANALYSIS

3.1 ADS Synthesized Netlist View

The RTL netlist and the technology netlist generated by the Compile and Synthesize phases of ADS can be viewed and analyzed using the View RTL Schematic and View Technology Schematic functions.

The RTL netlist can be viewed by clicking the View RTL Schematic toolbar button , and the view interface is shown in Figure 3-1. The toolbar buttons at the top are the functions supported by View RTL Schematic.

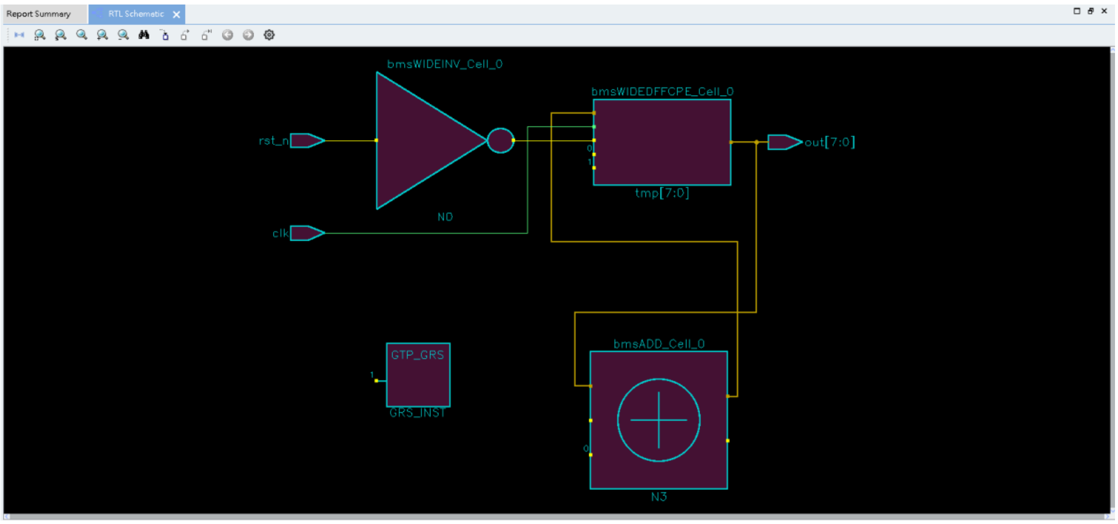


Figure 3-1 RTL Netlist

Design Browser: Clicking the toolbar button opens the Design Browser window, which displays the nets, ports, leaf cells and instances of the net list, and highlights the structures corresponding to the selected objects in the Design Browser in the net list.

Fit: fits the image; clicking the toolbar button scales all objects in the current view according to their outer borders, so that all objects can be displayed and fill the entire view.

Fit Selection: fits the selected objects; clicking the toolbar button scales all selected objects in the current view according to their outer frames, so that all selected objects can be displayed and fill the entire view.

Zoom In: clicking the toolbar button activates an interactive operation where the user can hold down the left mouse button and move around the view to draw a rectangle, then release the mouse button to scale the rectangle appropriately so that all objects within it are displayed and fill the view. Zoom In is a continuous interactive operation and requires Esc to exit.

Zoom In By 2: clicking on the toolbar button zooms the object in the current view twice as large, while reducing the viewable area.

Zoom Out By 2: clicking the toolbar button zooms the object in the current view twice as large and expands the viewable area.

Find: click the toolbar button to search for the object.

Descend: lower level display; click the toolbar button to activate Descend operation, then move the mouse to an instance object, when the cursor graphic becomes hand-shaped, it means you can enter the lower level design unit, click the left mouse button to enter; or select an instance object first, then right-click the menu, click the Descend item in the drop-down menu to enter the lower level design unit. The Descend item in the drop-down menu can also enter the lower level design unit.

Return: Top level display; if you are in a lower level design unit, click the toolbar button to go back one level.

Return to Top: Return to the top level display; if you are in a lower level design unit, click the toolbar button to return to the top level unit display.

Previous View: Previous view state; click the toolbar button to return to the previous view state; if the button is grayed out, it means the initial view state is already in place.

Next View: Next display state; click the toolbar button to display the next state; if the button is grayed out, it means the netlist is in the last display state.

Click the (View Technology Schematic) toolbar button to view the technology netlist, the view screen is shown below.

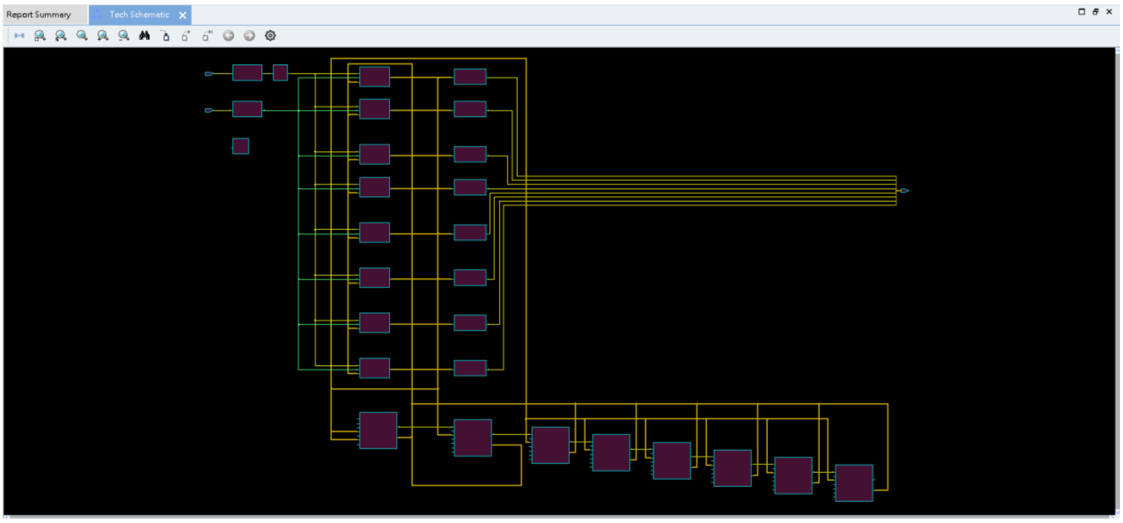


Figure 3-2 Technology Netlist

The function and operation of the top toolbar of View Technology Schematic is the same as View RTLSchematic.

Alt+left mouse button to drag the view, Alt+mouse wheel to move the view left and right, and Ctrl+mouse wheel to zoom in and out.

3.2 Design Browser Display

The design browser on the left side can display the resource information of the current view, as shown in the figure:

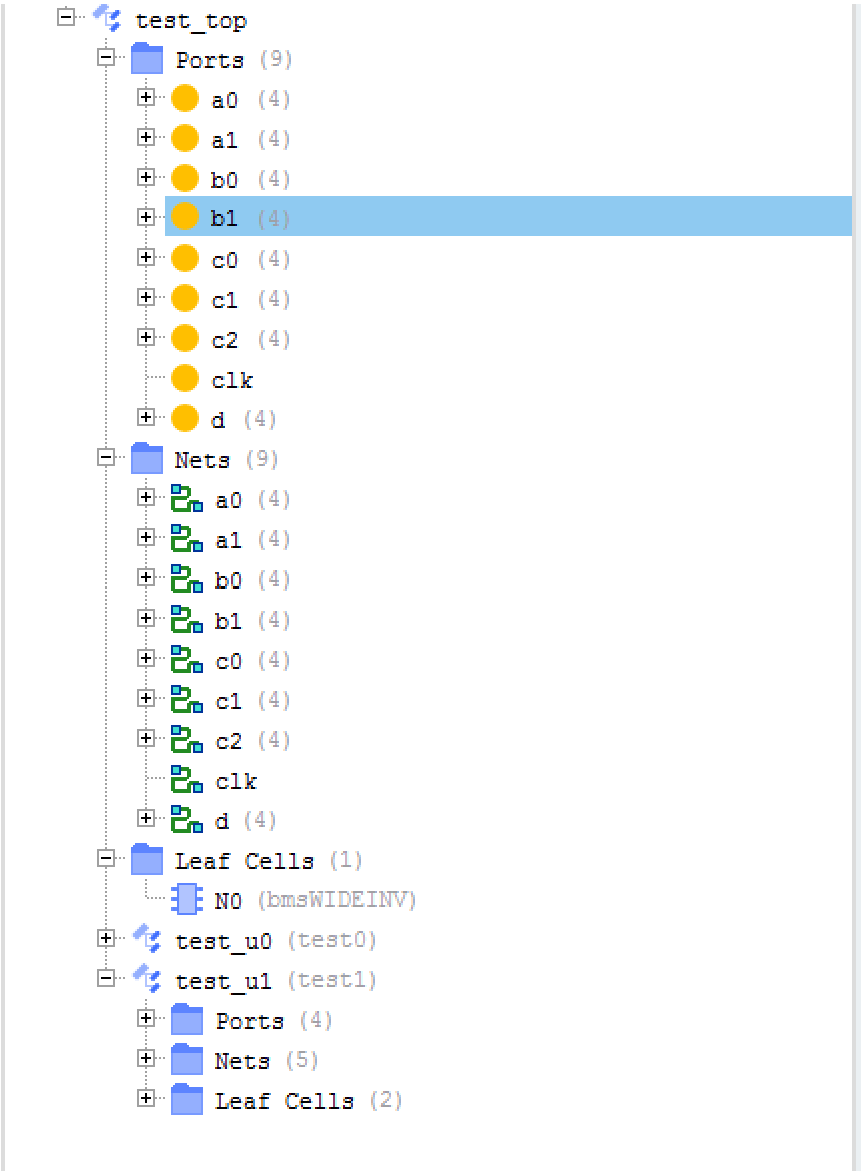


Figure 3-3 Design Browser

There are several right-click menus in the Design browser, which are listed below:

#	PRIMARY MENU	SECOND LEVEL MENU	DESCRIPTION
1	Zoom		Positioning the view to the selected object
2	Descend		Open the schematic of the selected object
3	Filter		Select the object to open the schematic
4	Filter In New window		Select the object to open the schematic in a new screen

5	Copy name	Copy name Copy Hierarchical name	Copy Name Copy layer name (without top layer)
6	View Obj In Source		Jump to the source file to view the object
7	View Module In Source		Jump to the source file to see the object model definition
8	Properties		Open the properties window for the selected object
9	Select Child Items	All	Select all objects under module
		All Ports	Select all ports under module
		All Instances	Select all Instances under module
		All Leaf Cells	Select all Leaf Cells under the module
10	Select Leaf Cells		Add only selected Leaf Cells under the current selected list, remove the selected Leaf Cell type.
11	Collapse all		Collapse all

3.3 Displaying Schematic Pages

Larger schematics are divided into multiple pages for display. The pages can be scrolled using the Page Up and Page Down keys on the keyboard. Clicking Page Down on the last page will return to the first page, and clicking Page Up on the first page will return to the last page.

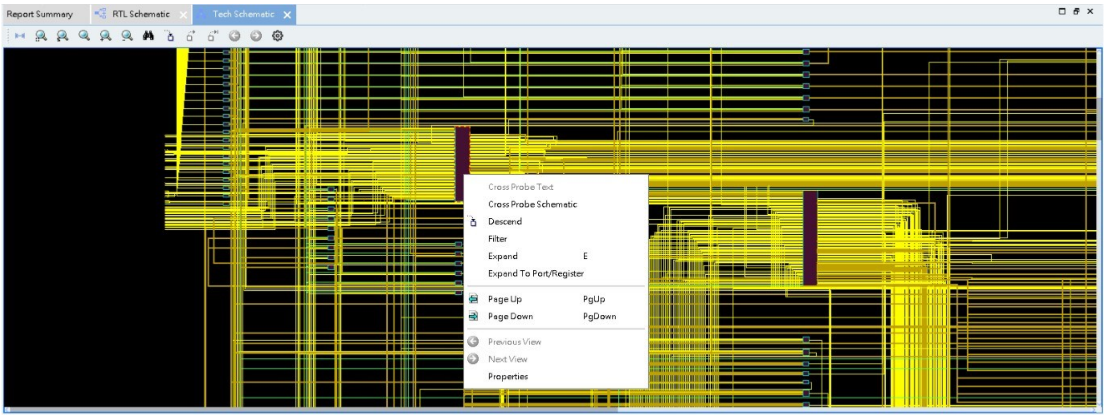


Figure 3-4 Paging Schematic Display

3.4 Find Object Function

There are often a large number of objects in the schematic, and the Find function allows you to search for a specific object. Click the button on the Schematic toolbar or press Ctrl+F to open the Find dialog box.

Enter the name of the object to be found in the Find what input box, which supports UNIX-style wildcards * and ? etc. For example, typing I* will match objects beginning with I; if you do not add any wildcards, a full match will be performed.

The following types of lookup objects are available: Instances to find unit instances directly, Cells to find their instances by cells, Groups to find groups composed of unit instances, Nets to find wire networks, and Ports to find ports. You can select one or more of them and quickly reset their status using the A and N buttons: A: All selected

A: All checked - all object types are selected.

N: None checked - deselects all object types

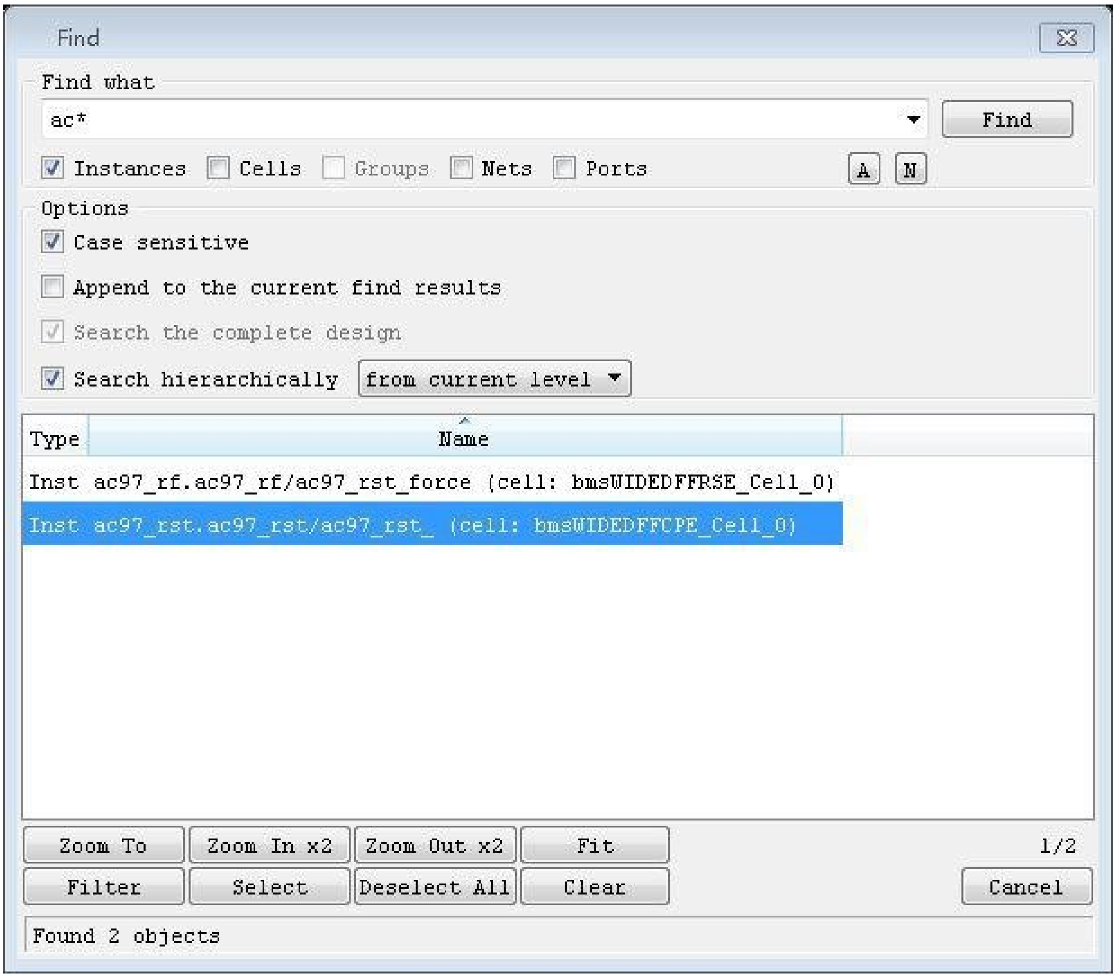


Figure 3-5 Schematic Search Dialog

Other options are

Case sensitive: case sensitive during the find process.

Append to current search results: When the search starts, the previous results are not cleared, and the newly found objects are appended to the results list.

Search the entire schematic: If the current SCM sheet is filtered, check this box to search the entire net list; otherwise, only the currently visible SCM sheet is searched.

Search Hierarchically: Search hierarchically, either top level starting from the top logical unit, or current level starting from the current logical unit. You cannot search for groups of cell instances in a hierarchical search.

Type in the Find what input field or click the Find button to start the search, and the results will be displayed in the list. Once the search is complete, you can move, zoom, and position the items in the list of results. The easiest way is to double-click on an object in the list to jump directly to the object's location and highlight it with a flash. There are also a few buttons that make it easy to perform some operations.

Zoom: If multiple selected objects are in the same cell, you can jump to that cell and zoom in on those objects. If these objects are on different pages in the same cell, a dialog box appears first, allowing you to select a page. You can also double-click an item in the result list to jump directly.

Zoom In x2: Zoom in twice to display. You can click on an object without selecting it to zoom in twice in the current window.

Zoom Out x2: Zoom out twice. Click without selecting to zoom out twice as large as the view in the current window.

Fit: Zooms the view in the current window to fill the entire window.

Filter: Filter the objects selected in the result list to display them in a different schematic. Note that these objects must be in the same design cell.

Select: If multiple selected objects are in the same cell, you can jump to that cell and select those objects.

Deselect All: Deselects all objects in the current window.

Clear: Clears the current search result.

Click the Cancel button or the dialog's Close button to close the dialog. If the search is not complete at this point, the user will be asked if they want to cancel the search.



Figure 3-6 Interrupt Finder

3.5 Schematic Preferences Settings

The user can set the display mode of the schematic through the preferences, etc. Clicking the (Preferences) button on the toolbar brings up a window with the following settings.

The Display section mainly controls some properties related to the schematic display content, etc., as shown in the figure.

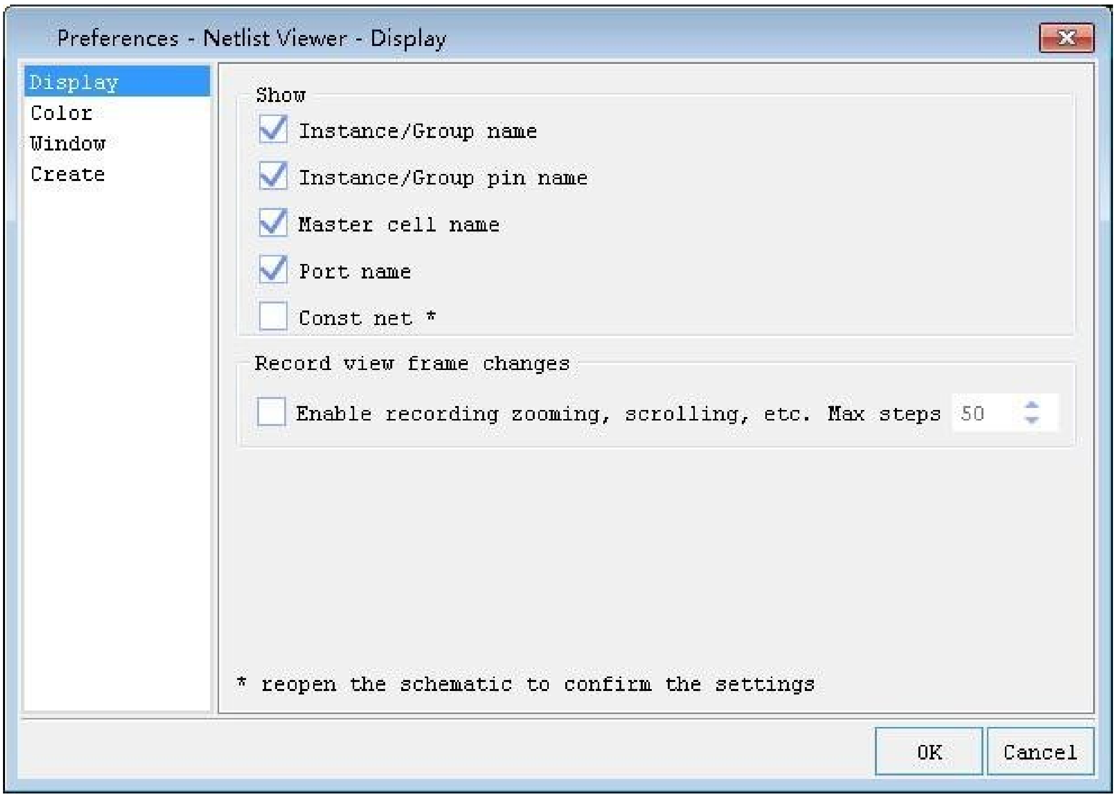


Figure 3-7 Schematic of the Preferences Settings dialog box Display section

- Instance/Group Name: Select whether to display the name of the entity or group.
- Instance/Group Pin Name: Select whether to display the pin name of the entity or group.
- Master cell name: Whether to display the cell name of the entity/group.
- Port name: Whether to display the port name.
- Const net: Whether to display the constant line net, such as VCC/GND, etc.
- Enable recording zoom, scroll: When there is zooming or scrolling in the drawing area, record these operations so that the previous operation can be repeated via Previous View / Next View. If this option is enabled, you can set the number of steps to be recorded.

The Color section controls the color of the schematic as it is drawn, as shown in the figure. Double-click one of the items in the list to select a new color scheme. Click the Reset button to restore the last saved color scheme or to return to the default white or black background color scheme.

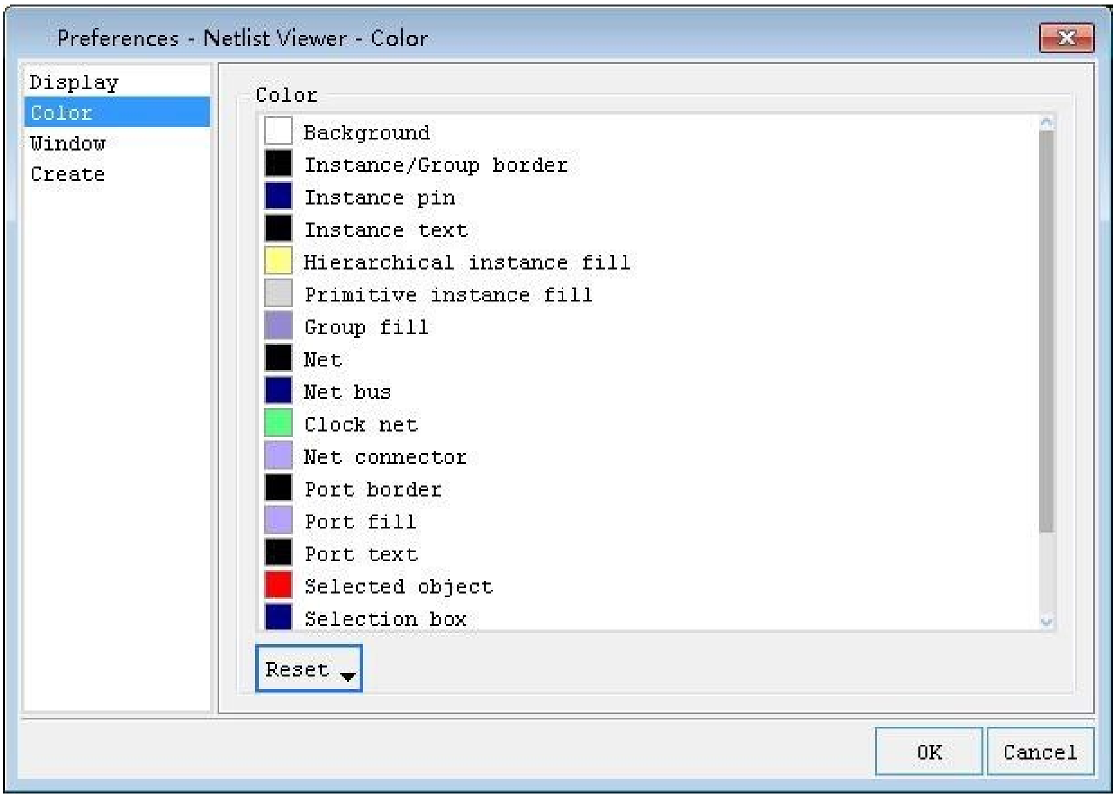


Figure 3-8 Schematic Preferences Settings dialog Color section

The Window section controls some of the controls of the Schematic window, as shown in the figure.

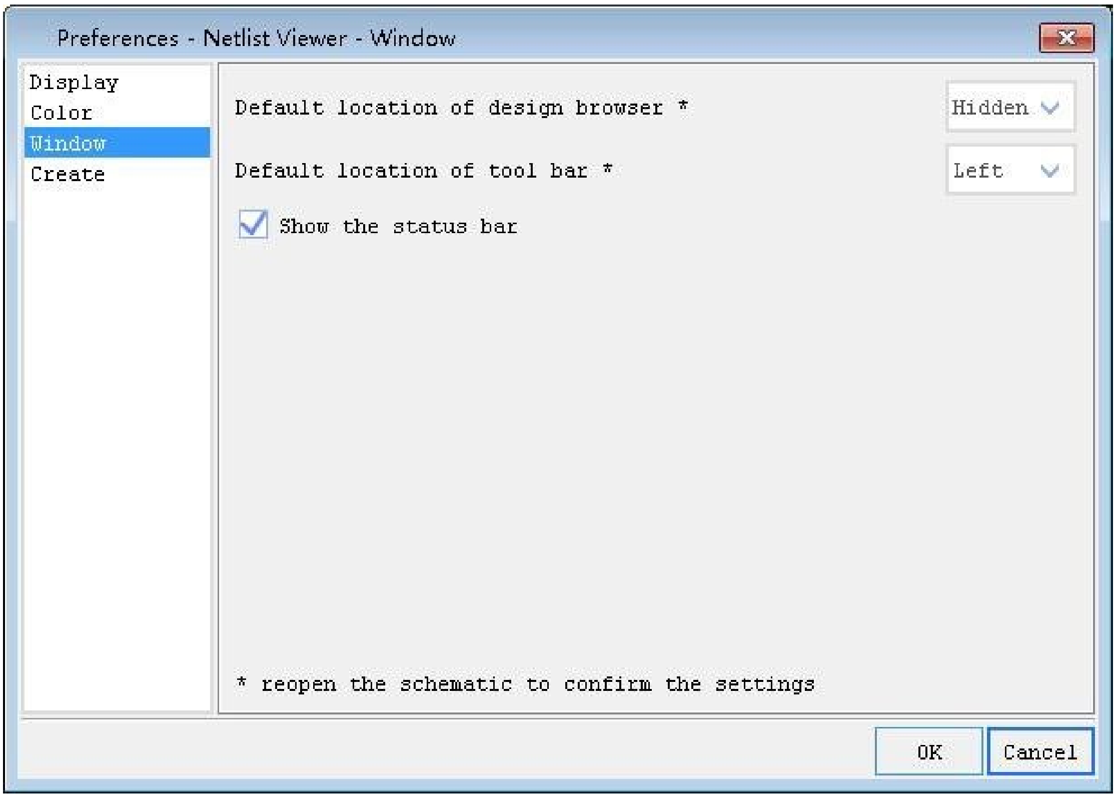


Figure 3-9 Schematic of the Preferences Settings dialog box section

- Default Design Browser Location: The default location of the design browser.
- Default Toolbar Location: the default location of the toolbar.

- Show Status Bar: whether to show the status bar at the bottom of the window.

The Create section controls some of the properties of the schematic generation process, as shown in the figure.

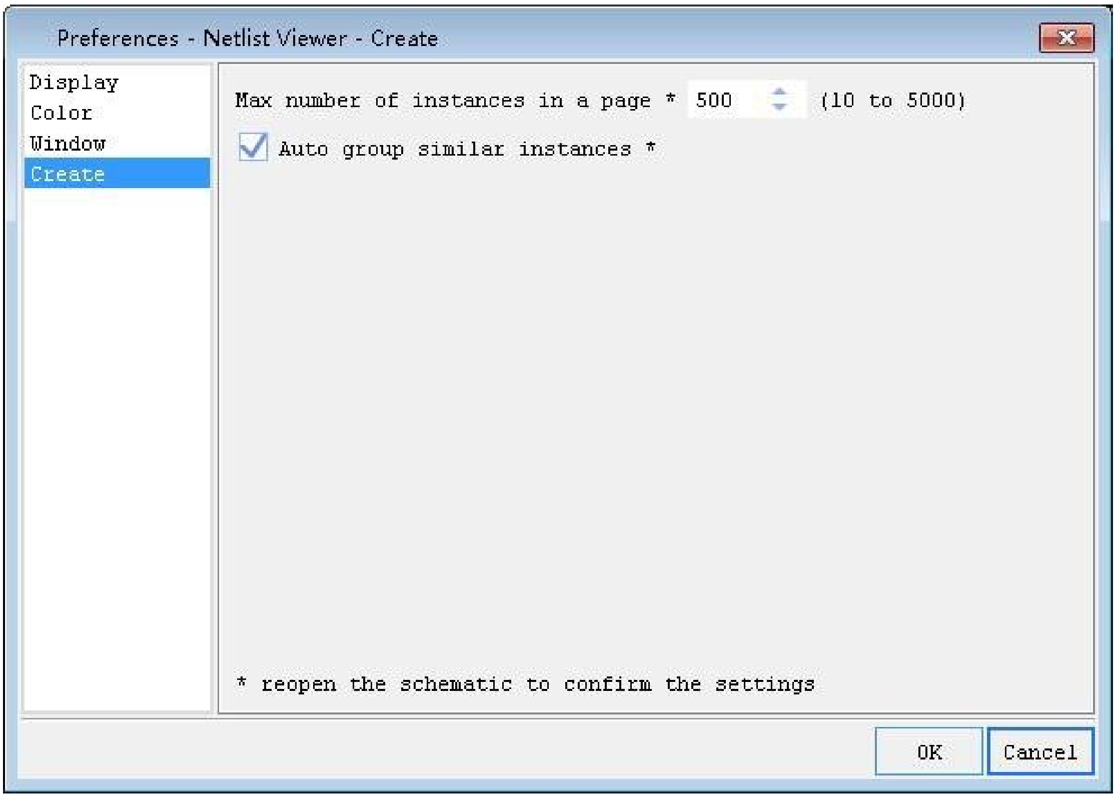


Figure 3-10 Schematic Preferences Settings Dialog Create Section

Maximum number of instances on a page: Sets the maximum number of entities that can be placed on each schematic page.

Automatically group similar instances: Automatically group entities with the same units and similar names.

3.6 RTL and Technology View Right Click Menu Introduction

After highlighting or selecting an object in the schematic, clicking the right mouse button brings up a pop-up menu for performing some operations. The contents of the menu vary depending on the object type and state. The actions defined in the menu are listed below:

View Object in Source: Schematic to text location, open the corresponding source file and locate it to the corresponding line.

View Object in RTL/Tech View: interactive positioning of RTL view and technology view; if an object is selected in RTL view, it can be positioned in technology view and the corresponding object is highlighted, and vice versa.

Edit Attributes: open UCE window to set attributes; this option is a function of RTL Schematic, Technology Schematic does not have this option.

Descend: you can go down to the selected instance or group object to see its details.

Filter: Displays the selected instance object in a separate window.

Filter in New Window: Displays the selected object separately in a new Schematic view.

Expand: Expands the objects connected to this instance.

Expand to Port/Register: Expands the objects connected to this instance to port/register and highlights them.

Page Up: Scrolls forward one page; for designs with multiple pages, click the Page Up button directly or right-click and select Page Up to scroll forward one page and restart from the last page when you reach the first page.

Page Down: Scroll backward; for multi-page designs, click the Page Down button directly or the right-click menu item Page Down to scroll backward, starting at the first page when you reach the last page.

Flatten: Flattens the hierarchy of the current schematic.

Copy Name: Copies the name of the currently selected object to the clipboard, or the name of the currently opened cell if no object is selected.

Show Connectivity: Highlight the net/port/instance to which the selected object is connected.

Properties: View the properties of the currently selected object.

Save as PDF...: Supports exporting and saving RTL or technology views as PDF files, you can select the range of views to export, export page size, page orientation, and the default export path is the same level directory as the project file.

3.6.1 Schematic Filtering

ADS supports filtering of single or multiple objects, and the way to select multiple objects is Shift+Left mouse button.

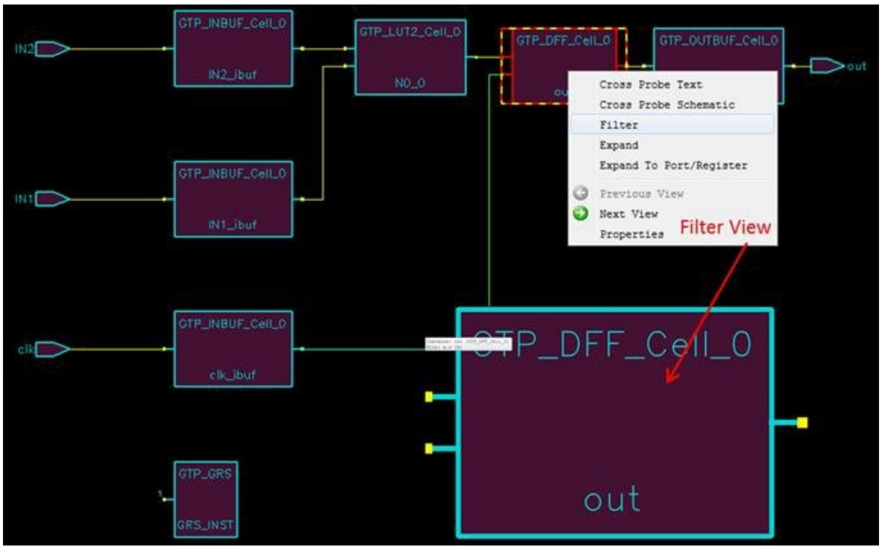


Figure 3-11 Filter Object Schematic

3.6.2 Expand and Expand to Port/Register

The Expand function expands the objects connected to the selected object and highlights the corresponding objects in the border. As shown in Figure 3-12, select the instance of N0_1, right-click and select Expand from the drop-down menu, and the border of the instance connected to N0_1 will be highlighted as shown in Figure 3-13.

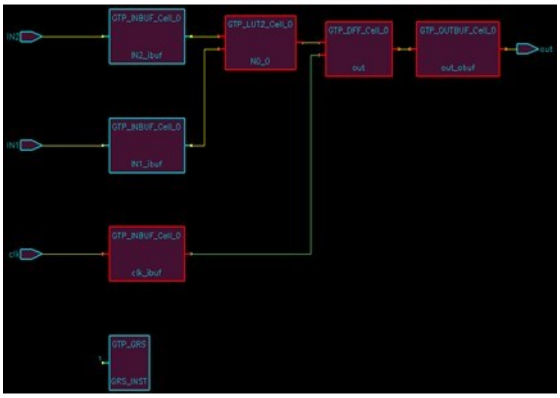


Figure 3-12 Schematic diagram object Expand

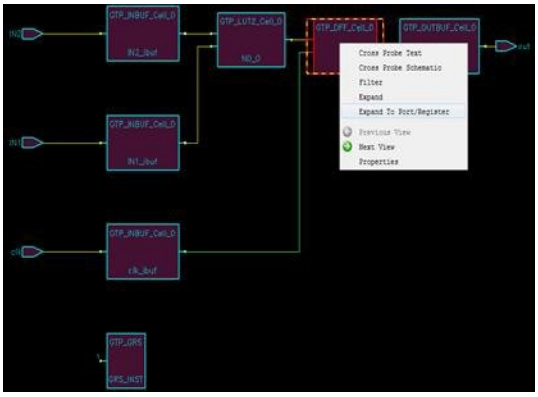


Figure 3-13 Schematic diagram object Expand

The Expand to Port/Register option is also available for net list expansion.

Expand to Port/Register highlights the boundary of the instance connected to the selected object and expands it to the port and register.

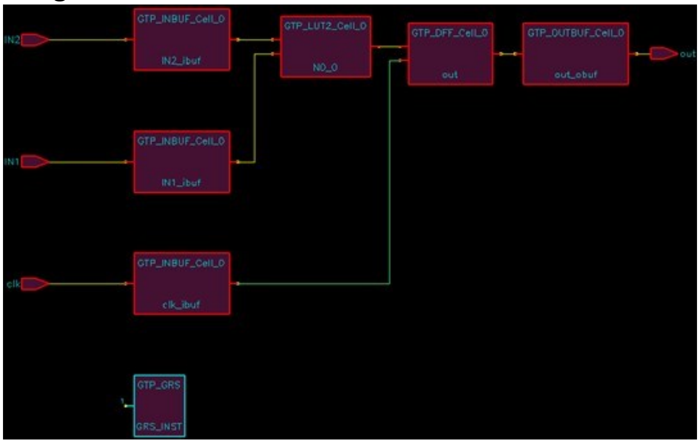


Figure 3-14 Expand to Port/Register

3.6.3 Flatten

When the current schematic is flattened, the original instances with hierarchies are replaced by the lowest level netlist. You can choose whether or not to flatten these instance groups (see below); note that even if you choose not to flatten instance groups, whether or not these groups are still generated is controlled by the Auto group similar instances option in Settings.

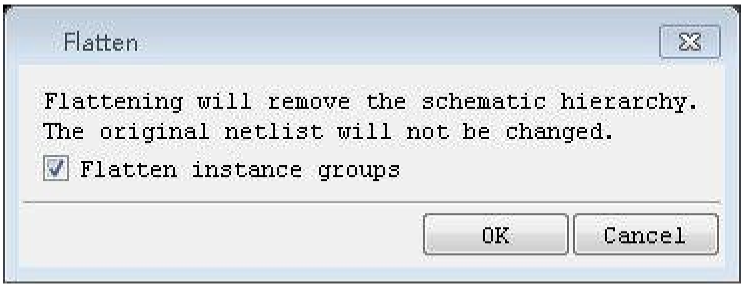


Figure 3-15 Flatten

3.6.4 View Properties

Move the mouse, select an object in the view, right-click and select Properties from the drop-down menu to display the following dialog box：

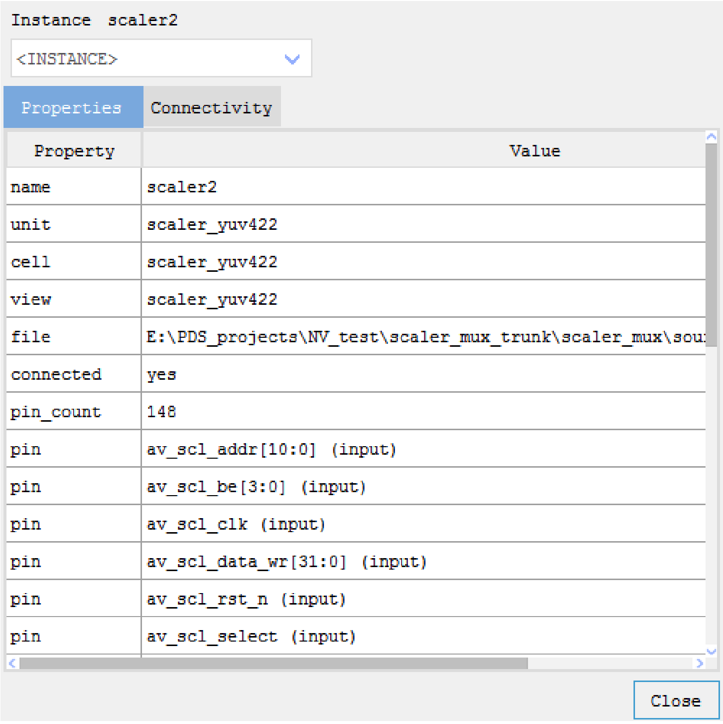


Figure 3-16 Properties dialog box

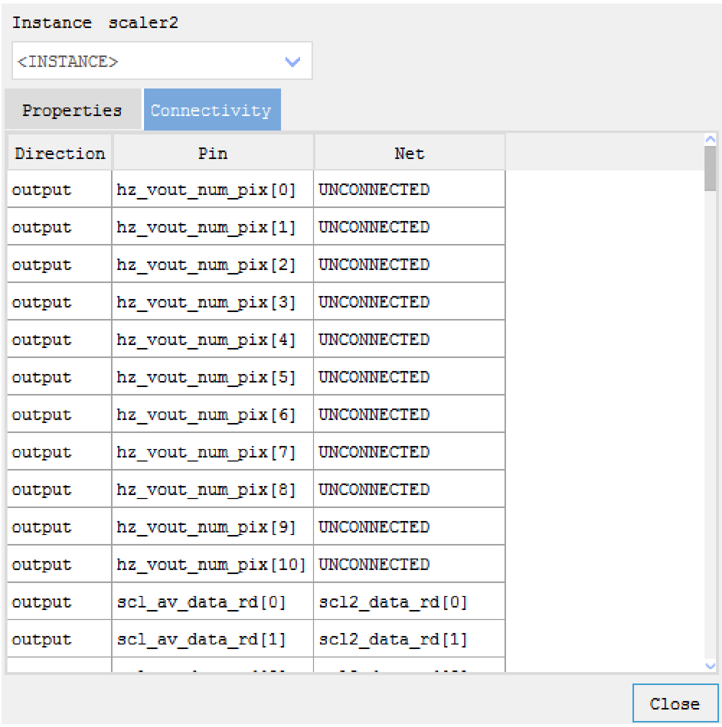


Figure 3-17 Connectivity dialog box

The above dialog allows you to view the properties or connection status of an object. For instance objects, you can also choose to view their port, and for bus-type net or port objects, you can choose to view a subset of them.

You can click or double click with the mouse:

#	INTERFACE NAME	DESCRIPTION
1	Double click on Net in the list object	Locate and flash the Net in view
2	Double click on a Pin in the list object	Locate and flash the Pin in the view Belonging Instance
3	Double-click the Instance in the list object	Locate and flash the Instance in the view
4	Double-click the Port in the list object	Locate and flash the Port in the view
5	Double-click the Group in the list object	Position and flash the Group in the view
6	Single choice, Multiple choice	Supported (via Ctrl or Shift multiple choice) , only the form is selected

There are several right-click menus in the Properties window, which are listed below:

#	Features	Description
1	Focus	Orient the view to the selected object and blink (same double-click Function)
2	Clear focus	Cancel interface flashing
3	Select	Selected object, table, view, Design Browser Both selected
4	Deselect All	Uncheck the interface
5	Filter	Filter selected objects
6	Copy	Copy name
7	All	Select all cells in the current property window

The Properties window supports column sorting, as follows:

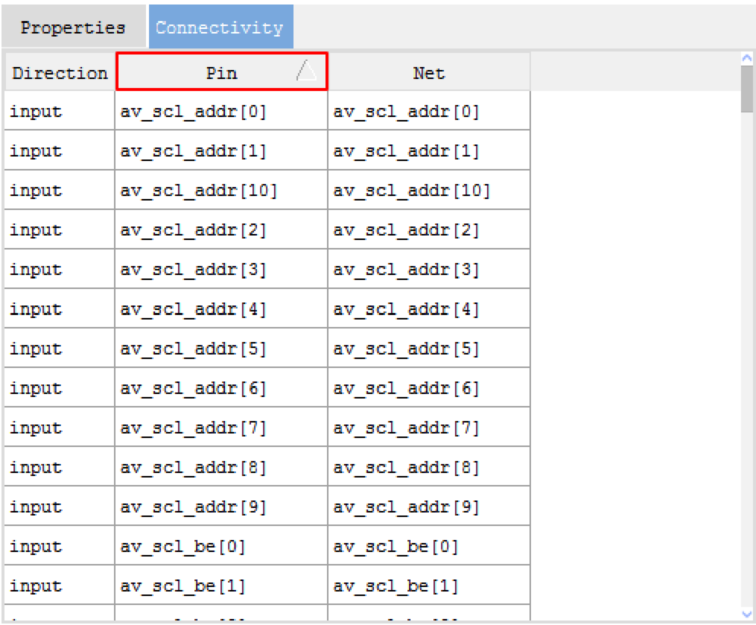


Figure 3-18 Column Sorting Support

View operations can be performed while the property window is open (note: only one property window can be open at a time for a NV)

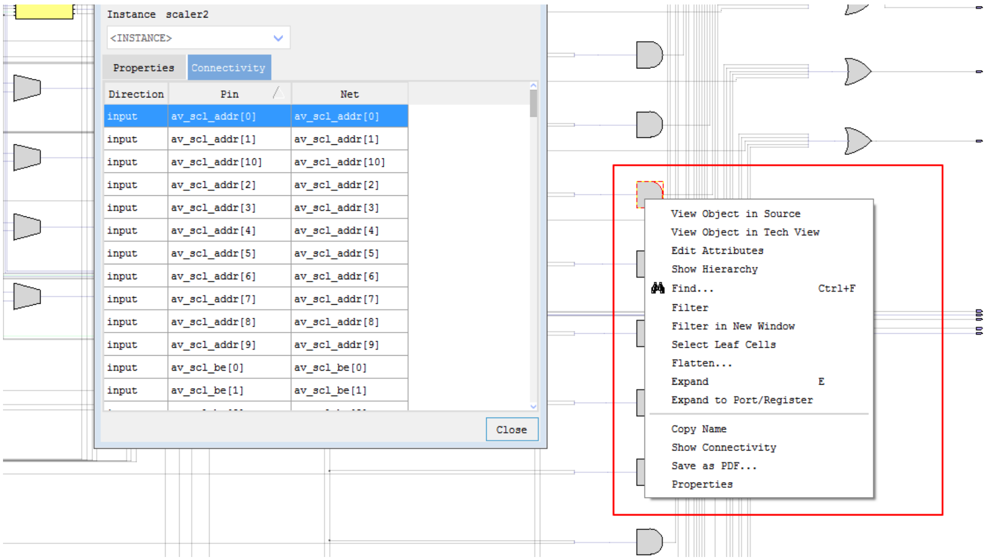


Figure 3-19 Open the Properties window while operating the view

3.7 Interaction between Schematic View and Verilog Source Code

After selecting an object in the Schematic View, right-click and select Cross Probe Text from the menu to open the corresponding source code in the text editor and locate the corresponding description line of the object in the source code. The corresponding description is highlighted in the source file.

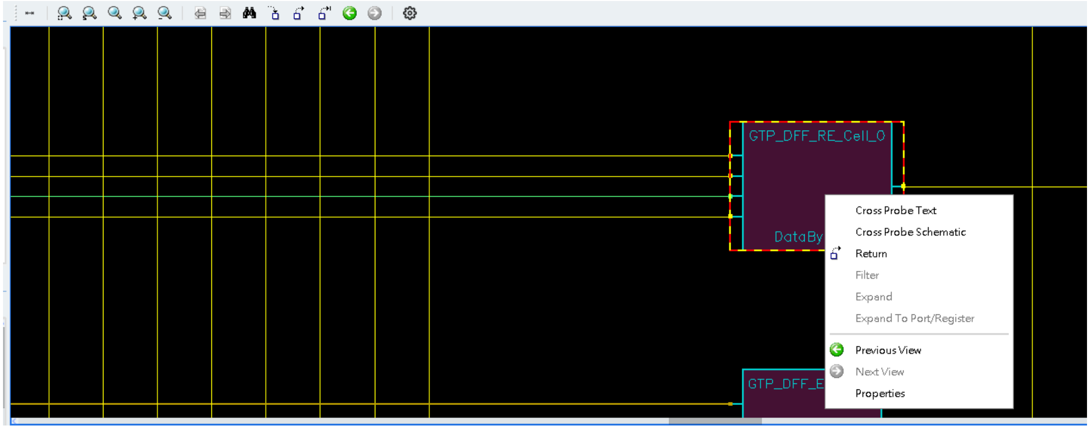


Figure 3-20 RTL Netlist and Text Interactive Positioning

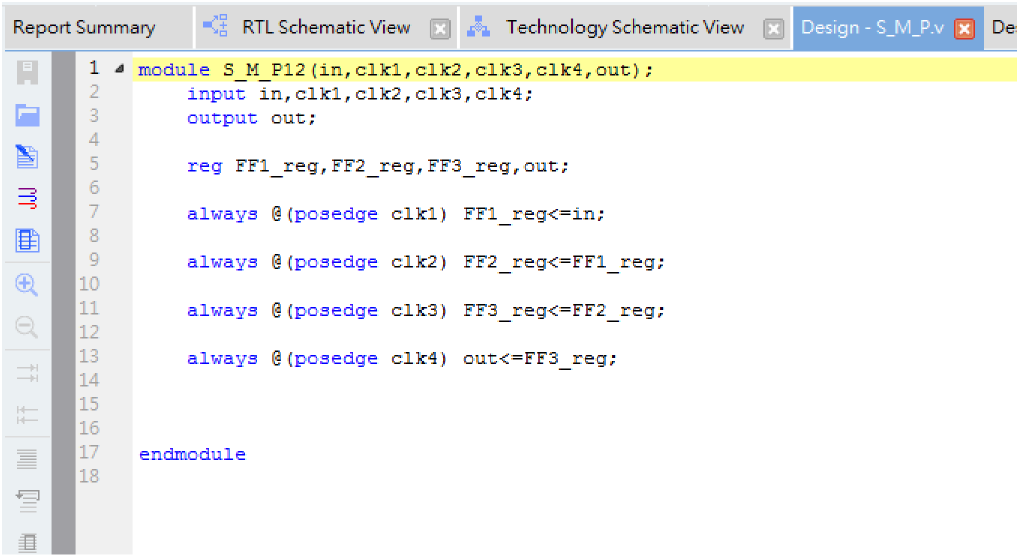


Figure 3-21 Verilog file display result

3.8 Cross Probe between RTL Schematic View and Technology Schematic View

After selecting an instance in the schematic, right-click and select Cross Probe Schematic from the menu to cross probe between the RTL view and the Technology view.

In the figure below, select the instance in the RTL view, right-click and select Cross Probe Schematic to automatically switch to the Technology Schematic view and highlight the corresponding instance. Conversely, the same operation in the Technology Schematic view will jump to the RTL Schematic view and highlight the corresponding instance in the outer frame.

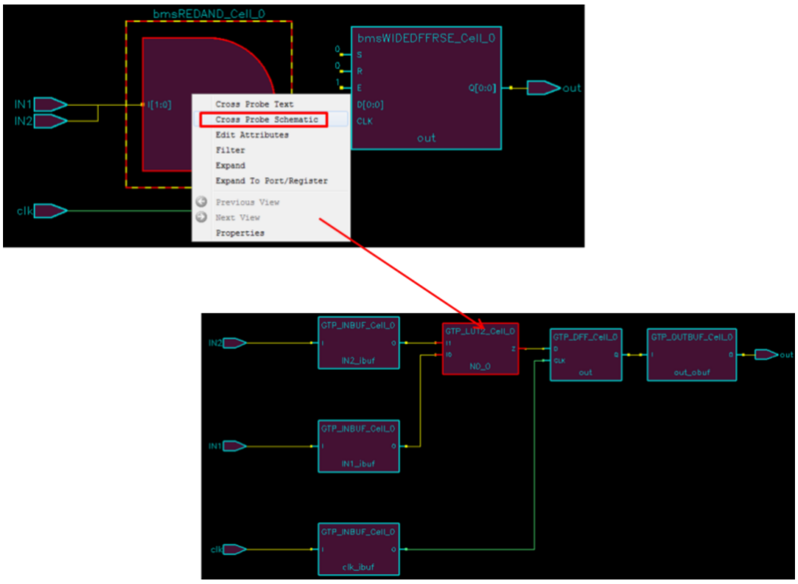


Figure 3-22 RTL and Technology Schematic Interactive Positioning

3.9 Critical Path Jump to Schematic

In the Timing Report, select any critical path, right-click, and select View Timing Path In Schematic from the right-click menu to jump to the schematic to view the timing path.

For example, in the Synthesize Timing Report main interface, select the Start Point/End Point object of the critical path, right-click, and select View Timing Path In Schematic from the right-click menu, as shown in the figure below.

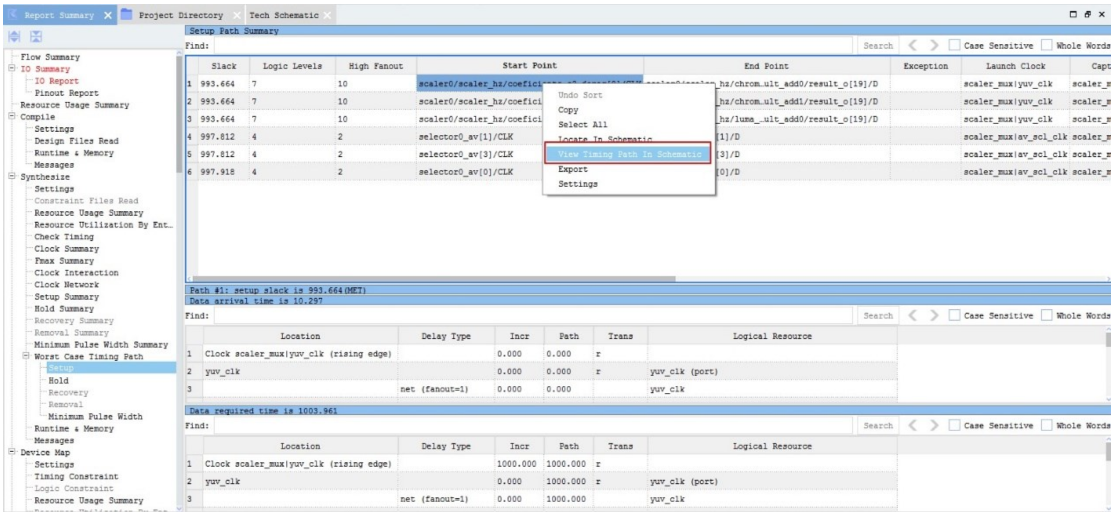


Figure 3-23 Right-click menu option View Timing Path In Schematic

It will jump to the main interface Tech Schematic to display the schematic diagram of the timing path, as shown in the figure below.

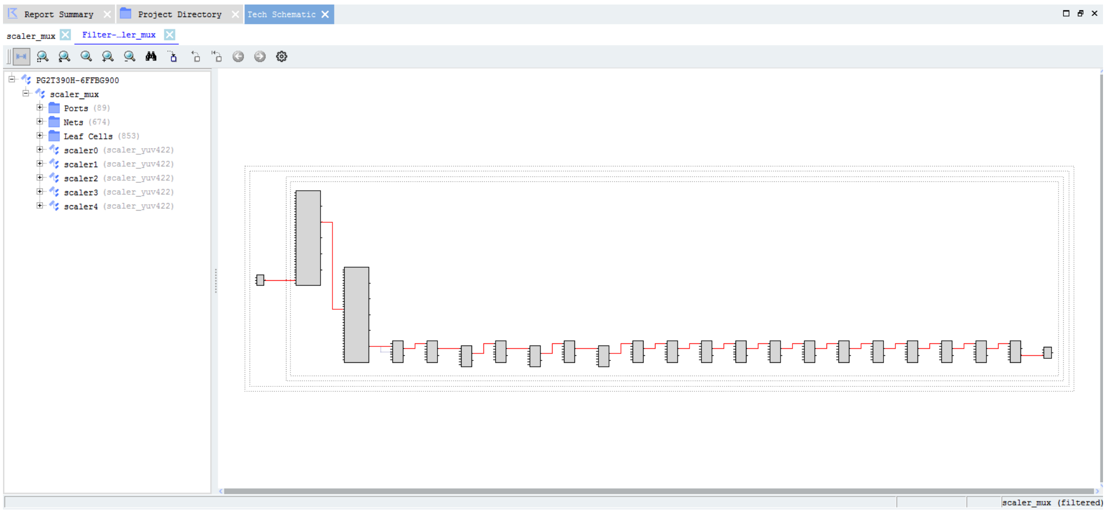


Figure 3-24 Going to the Tech Schematic to View the Timing Path

As in the TA Component Timing Report, select the Start Point/End Point object of the critical path, right-click and select View Timing Path In Schematic from the shortcut menu, then jump to the Design Schematic to display the timing path schematic, as shown next.

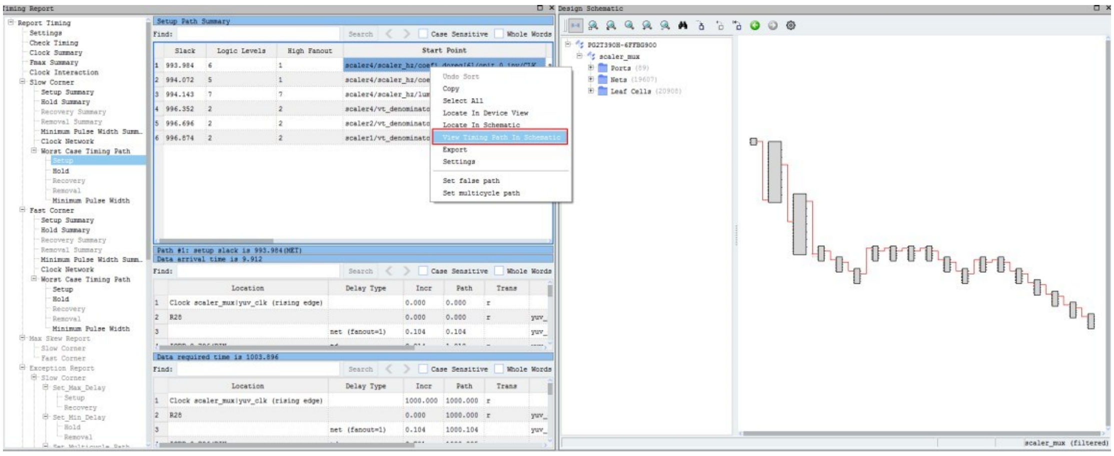


Figure 3-25 Going to the Design Schematic to View Timing Paths

In the Timing Report, select any Logical Resource object on the critical path, right-click, and select Locate In Schematic from the right-click menu to jump to the schematic to view the object.

As in the main Synthesize Timing report interface, select the Pin/Port/Net object in the critical path, right-click, and select Locate In Schematic from the right-click menu, as shown below.

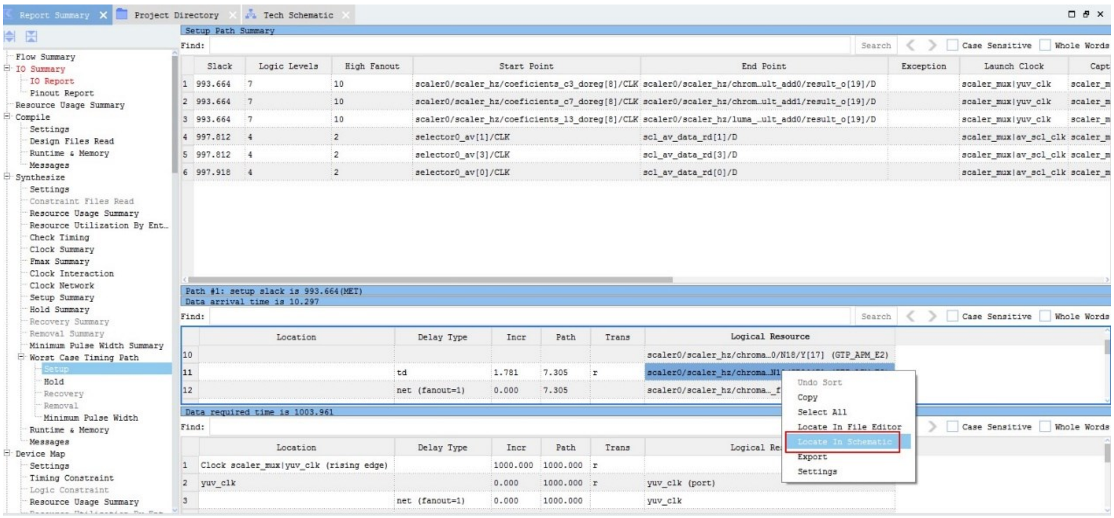


Figure 3-26 Right-click menu option Locate In Schematic

The main screen will jump to Tech Schematic to locate the object, as shown in the figure below.

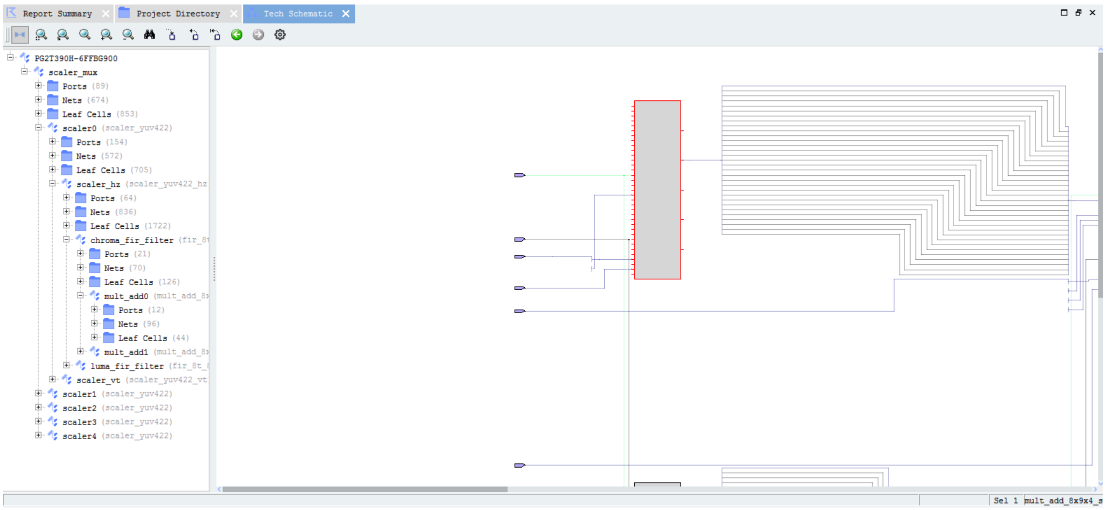


Figure 3-27 Jump to Tech Schematic to Locate and Select Object

As in the TA Component Timing Report, select the Pin/Port/Net object on the critical path, right-click and select Locate In Schematic from the right-click menu, then jump to the Design Schematic to select the object, as shown in the figure below.

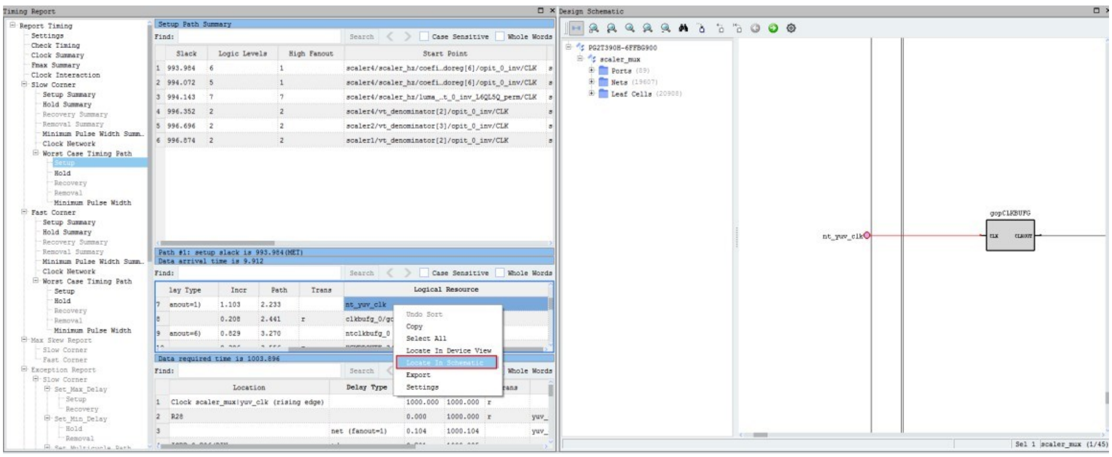


Figure 3-28 Jump to Design Schematic to locate the selected object

4. SHELL COMMAND TO PERFORM ADS SYNTHESIS

4.1 Flow of ADS logical synthesis using shell commands

set_arch -family Compact [-device device -package package -speedgrade speedgrade]

add_constraint -logic -fdc fdc_file

add_design -verilog design_file

compile [-include_path include_path -top_module modulename]

synthesize -ads

set_arch family can be set to Compact and cannot be defaulted; the default setting will be used for the part in [].

The add_constraint command completes the addition of a constraint file, which is of type .fdc.

add_design-verilog command completes the addition of the verilog file to be synthesized.

compile command completes the compile phase of ADS, -include_path and -top_module can be defaulted.

The synthesize-ads command completes the synthesize phase of ADS, and generates and technology netlist file (.vm) after map and optimization, which can be used to do verification work with third-party verification tools.

The following is an example of a script that implements ADS logical synthesis by writing shell commands to a file such as MyScript.tcl (extension .tcl) and executing the command pds_shell -file MyScript.tcl to complete the ADS synthesis.

set_arch -family Logos -device PGL50H -package FBG256 -speedgrade sg7

add_constraint -logic -fdc design.fdc

add_design -verilog a.v

compile

synthesize -ads

4.2 Compile and Synthesize -ads Option

The compile command completes the compile phase of the ADS, which has a corresponding option for the user to set. compile option and its corresponding description are as follows：

Option	Description
-include_path	include_path is used to set the include path of the compiler, and the option must be followed by an argument; If there is only one include path, give the pathname as an argument to the option; if there are multiple include paths, you must enclose the list of include paths in {}.
-top_module	Specify the name of the top-level module, this option must be followed by a parameter, the parameter is the module name; if this option is not set, ADS will automatically select the module that has not been instantiated as the top-level module, if there are multiple modules in the input file that have not been instantiated, ADS will select the first module that has not been instantiated as the top-level module according to the alphabetical order.
-system_verilog	The default Verilog is set to System Verilog after the option is set, and no parameters are needed after this option.
-allow_duplicate_modules	If unchecked, an error is reported if a module with the same name exists in the design. If this option is checked, the module with the same name can be successfully compiled and the software will overwrite the previous module with the latter module in the order of module parsing, no parameter is required after this option.
-multi_file_compilation_unit	Switch that controls whether variables, macros, etc. defined in the current RTL file can be validated in other RTL files, switch attributes, no arguments required.
-loop_limit	Sets the upper limit on the number of loops in the for statement. The option must be followed by a parameter that is the upper limit of the loop and must be greater than 2000 before it will be written.
-fsm_compiler	The encoding option for the FSM infer. The values after this option can be auto, off, one_hot, gray, sequential, safe, safe, one_hot, safe, gray, safe, sequential, safe, original
-defined_parameters	The parameter values set in the options override the top-level parameter values in the design. Use a space to separate parameter name and parameter value, if there are multiple parameters, different parameters are also separated by a space, all parameters must be enclosed in {}, for example {ADDR_WIDTH 8 DATA_WIDTH 4 ASYNC "TRUE"}.
-compiler_directives	Used to set macro definition parameters, use '=' between macro definition name and macro definition value, use space to separate multiple macro definition statements, if there are multiple macro definition parameters, you must enclose all macro definition parameters in {}, such as {ADDR_WIDTH=8 DATA_WIDTH=4 ASYNC}.
-work	Specify the name of the working library, followed by a parameter that is the name of the working library that the user wants to set.

The synthesize -ads command completes the synthesize phase of the ADS process, which has a corresponding option for the user to set. synthesize command option and corresponding description are as follows:

Option	Description
-fanout_guide	Set the maximum value of fanout, the value after option is an integer variable
-rw_check_on_ram	Automatic read/write check insertion for RAM, switch property, no arguments required
-disable_io_insertion	Disable I/O Insertion , switch property, no parameters required
-enable_prepacking	LUT6D pack switch, switch attribute, no parameter required
-resource_sharing	Resource sharing switch. the parameter after option can be set to true/false, if set to true, resource sharing is performed, if set to false, resource sharing is not performed.
-min_controlset_size	Sets the minimum value of FF with the same control port, the value after the option is an integer variable.
-retiming	Retiming switch, can optimize timing by moving registers in synthesis, switch property, no parameters required
--pipelining	Pipelining switch, can optimize area by moving registers in synthesis, switch property, no parameters required
-max_path	Sets the number of worst timing paths reported in the integrated timing report, the value after the option is an integer variable.
-nworst	Specify the maximum number of paths that can be stored for each endpoint reported in the integrated time analysis, the value after the **option** is an integer variable.
-path_logic_level_gt_than	The logic levels of each setup/recovery timing path reported in the integrated timing analysis must be greater than the set value, and timing paths less than this value are not displayed; the value after the option is an integer variable.
-frequency	For infer clock, if the correct clock frequency cannot be calculated based on the PLL parameters and the reference clock, this option is used to generate a clock with a frequency greater than 1.0000 before writing.
-use_clocked_period_for_unconstrain_io	Add IO delay constraints for ports where the user does not have IO delay constraints to improve coverage of timing constraints, switch attribute, no parameters required

5. ATTRIBUTE REFERENCE

There are two ways to set an attribute, either in the HDL source file or in the fdc constraints file. fdc constraints are set at the synthesize stage, and if the attribute to be set is valid at the compile stage, the attribute must be set in the HDL source file.

Among the attributes supported by ADS, the only ones supported in the HDL source file are

syn_black_box

syn_looplimit

syn_keep

syn_noprune

syn_preserve

The user sets the attribute from the fdc file as follows

define_attribute <ObjectName> <AttributeName> <AttributeValue>.

The user sets the attribute in the HDL source file as follows

(Verilog-1995 meta-comments)

/*synthesis <AttributeName> = <AttributeValue> */ or

or (Verilog-2001 attribute)

(* <AttributeName> = <AttributeValue> *)

The synthesis attributes supported by ADS and their descriptions are listed below.

5.1 syn_allow_retiming

Specifies whether the register allows retiming optimization.

This attribute is mainly used to specify that some registers should not be retiming optimized if global retiming optimization is enabled through the option

syn_allow_retiming Values

Global	Object
Yes	Register

PROPERTY VALUE	FUNCTION
1	Retiming optimization is allowed, and this register can be optimized if global retiming optimization is enabled in the Synthesis configuration options.
0	Retiming optimization is not allowed, and the register is not optimized when global retiming optimization is turned on in the Synthesis configuration options.

Syntax Specification

FDC file	`define_attribute {i: instance} syn_allow_retiming {0}`
	Global properties:
	`define_global_attribute  syn_allow_retiming {0}`

Verilog	`object /* synthesis syn_allow_retiming = 1 */;`

Verilog Example

```
module test (a, rs, out, clk) ;
input [15:0] a;
input rs;
input clk;
output out;

reg [15:0] ar;


always @ (posedge clk)
  begin
      ar <= a;
  end


wire t;
assign t = |ar;


reg tr /* synthesis syn_allow_retiming = 0 */ ;

always @ (posedge clk)
  begin
      tr <= t;
   end


reg out;
always @ (posedge clk)
  begin
      if (rs)
          out <= 1;
   else
out <= tr;
end

endmodule
```

This property must be used with the Retiming option in the configuration options, i.e., the global Retiming option must be checked first. If the Retiming option is checked in the example and the property is not used before, retiming optimization is performed in the synthesis phase.

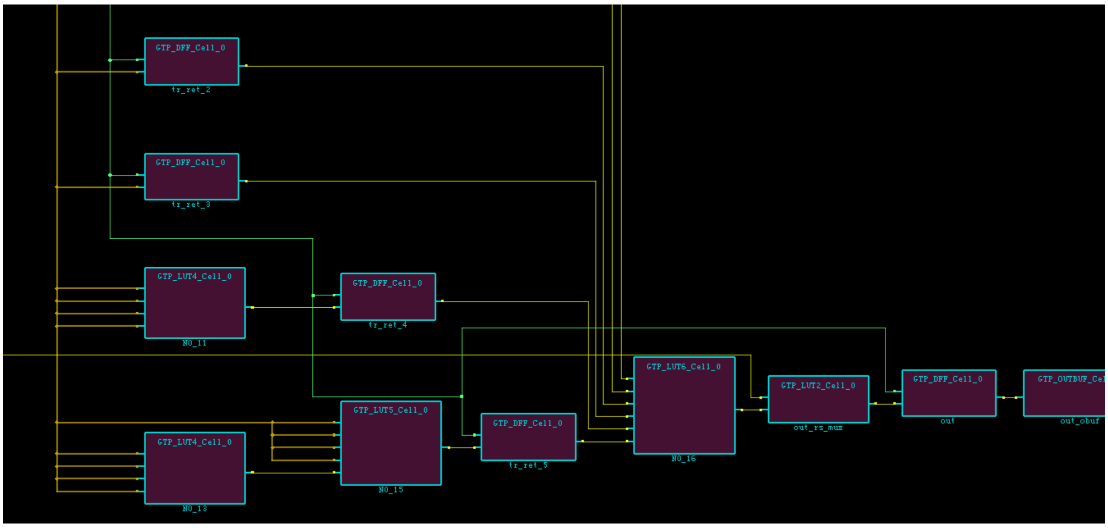


Figure 5-1 Checking Retiming for Optimization

With syn_allow_retiming = 0, no retiming optimization is performed even if the retiming option is checked, as shown in the figure; this property is mainly used to control local registers without retiming optimization.

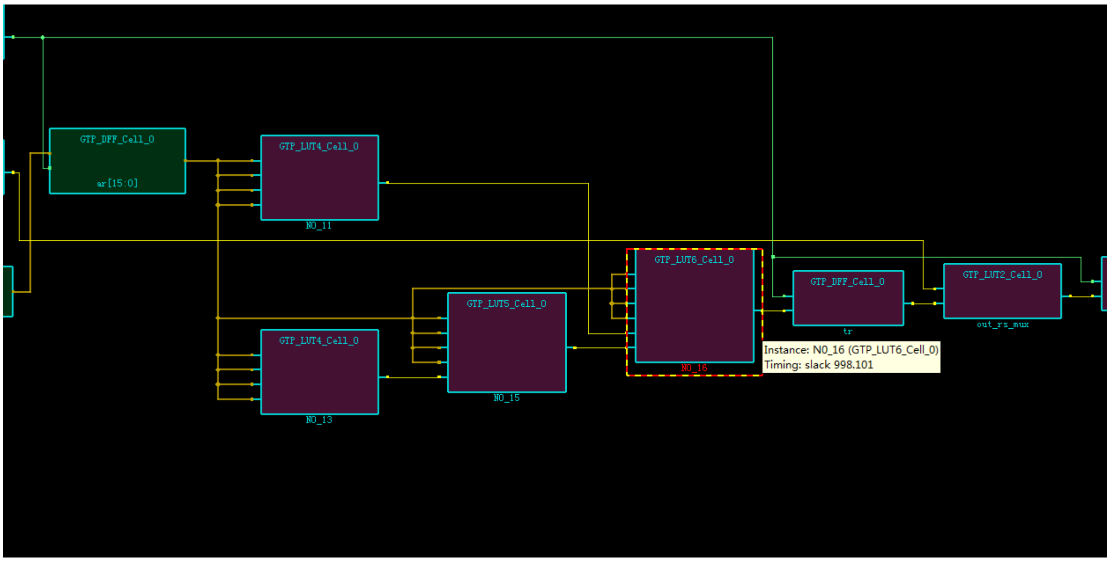


Figure 5-2 No Retiming optimization

5.2 syn_allowed_resources

If the used DRM resources exceed the set limit, the inferred ram is automatically mapped to distributed ram or registers, and the instantiated GTP ram is not affected. Global attributes only are supported.

syn_allowed_resources Values

GLOBAL	OBJECT
Yes	Top module

PROPERTY VALUE	FUNCTION
blockrams=%d	If the DRM resources used exceed the set limit, the excess inferred RAM is automatically mapped to distributed RAM or registers.

Syntax Specification

FDC file	`define_global_attribute {syn_allowed_resources} {blockrams=100}`
	or
	`define_attribute {v:top module} {syn_allowed_resources} {blockrams=100}`

Verilog	`object /* synthesis syn_allowed_resources = "blockrams=10" */;`

Verilog Example

```
module top# (

    parameter DATA_WIDTH = 10,
    parameter ADDR_WIDTH = 15
) (
    input CLK,
    input WE,
    input [ADDR_WIDTH-1:0] ADDR,
    input [DATA_WIDTH-1:0] DI,
    output wire [DATA_WIDTH-1:0] DO

)/* synthesis syn_allowed_resources = "blockrams=5" */;


sub sub1(CLK,WE,ADDR,DI,DO);


endmodule

module sub # (

    parameter DATA_WIDTH = 10,
    parameter ADDR_WIDTH = 15
) (
```

```
    input CLK,
    input WE,
    input [ADDR_WIDTH-1:0] ADDR,
    input [DATA_WIDTH-1:0] DI,
    output reg [DATA_WIDTH-1:0] DO

);


reg [DATA_WIDTH-1:0] mem [2**ADDR_WIDTH-1:0] ;


always @ (posedge CLK)

    if (WE) mem[ADDR] <= DI;

    else do <= mem[ADDR];


endmodule
```

Before this property is applied in the example, the combined inferred rams are all mapped to the DRM and the resources are reported as follows:

Mapping Summary :

Total LUTs: 0 of 243600 (0. 00%)
LUTs as dram: 0 of 75400 (0. 00%)
LUTs as logic: 0
Total Registers: 0 of 487200 (0. 00%)
Total Latches: 0

DRM36K/FIFO:
Total DRM = 10. 0 of 480 (2. 08%)

After applying syn_allowed_resources = "blockrams=5", up to 5 DRMs can be used in the project, the excess is automatically mapped to Distributed RAM and FF, and the resource summary can be seen in the resource report after synthesis as follows

```
Mapping Summary :

Total LUTs: 2862 of 243600 (1.17%)
     LUTs as dram: 2560 of 75400 (3.40%)
     LUTs as logic: 302
Total Registers: 5 of 487200 (0.00%)
Total Latches: 0

DRM36K/FIFO:
Total DRM = 5.0 of 480 ( 1.04%)
```

5.3 syn_black_box

Specify a module as black box

When synthesizing, you do not need to know how a module is implemented, but only its input and output interfaces, which can be set to black_box.

syn_black_box Values

GLOBAL	OBJECT
no	view

PROPERTY VALUE	FUNCTION
1	object is set to black_box
0	Not black_box , same Function as not setting
N/A	Same Function as setting to 1, object is set to black_box

Syntax Specification

Verilog	`object /* synthesis syn_black_box = 1 */;`
	
	`or object /* synthesis syn_black_box  */;`

Verilog Example

```
module top(clk, in1, in2, out1, out2);


input clk; input
[1:0] in1;
input [1:0] in2;
```

```
output [1:0] out1;
output [1:0] out2;

add          U1 (clk, in1, in2, out1);
black_box_add U2 (in1, in2, out2);

endmodule

module add (clk, in1, in2, out1);


input clk; input
[1:0] in1;
input [1:0] in2;

output [1:0] out1;
reg [1:0] out1;

always @(posedge clk) begin
      out1 <= in1 + in2;
   end

endmodule

module black_box_add(A, B, C) /* synthesis syn_black_box = 1 */;

input [1:0] A;
input [1:0] B;

output [1:0] C;

assign C = A + B;


endmodule
```

This property is visible inside the black_box_add module in the compile netlist before it is applied in the example, and is duplicated in the add module that is optimized during the synthesis phase.

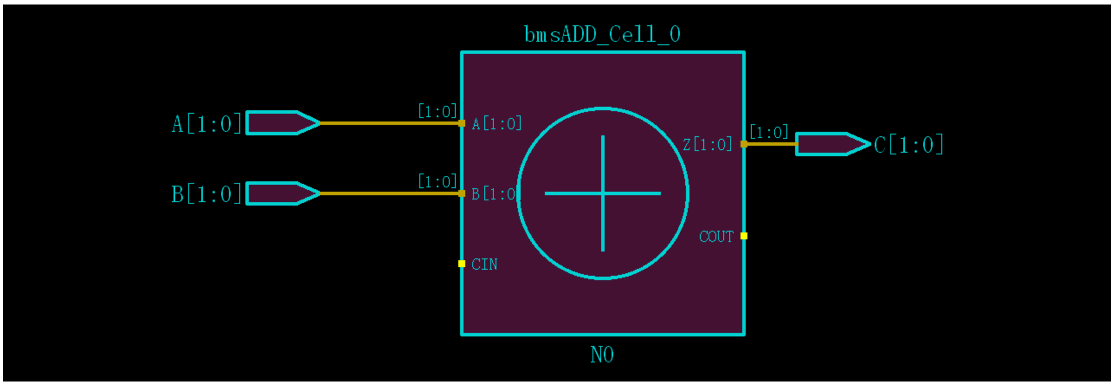


Figure 5-3 Compile Netlist

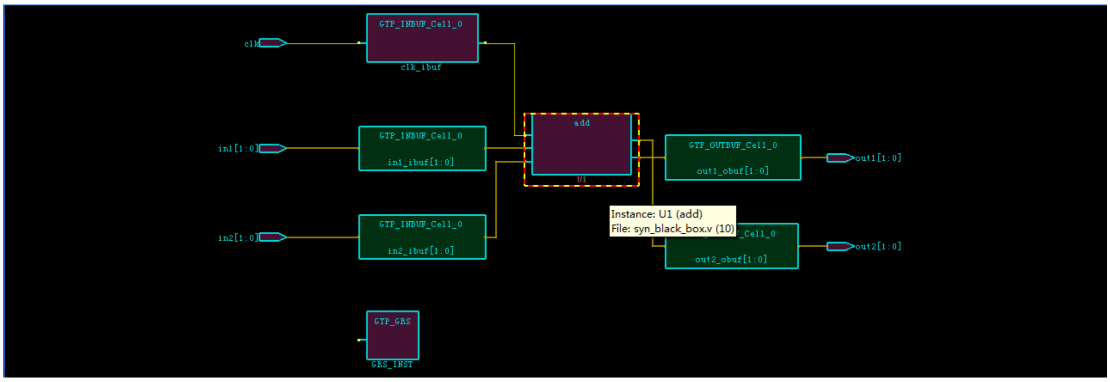


Figure 5-4 Synthesize Netlist

After applying syn_black_box = 1, the internal black_box_add module becomes a black box, only the port information is visible, the internal functions are not visible, so it is not optimized and the black box module is passed on.

Note: Since the black box cannot be flattened, an error is reported in the device map phase:

E: Flow-0088: Design instance U2 is not completely flattened or contains black boxes:

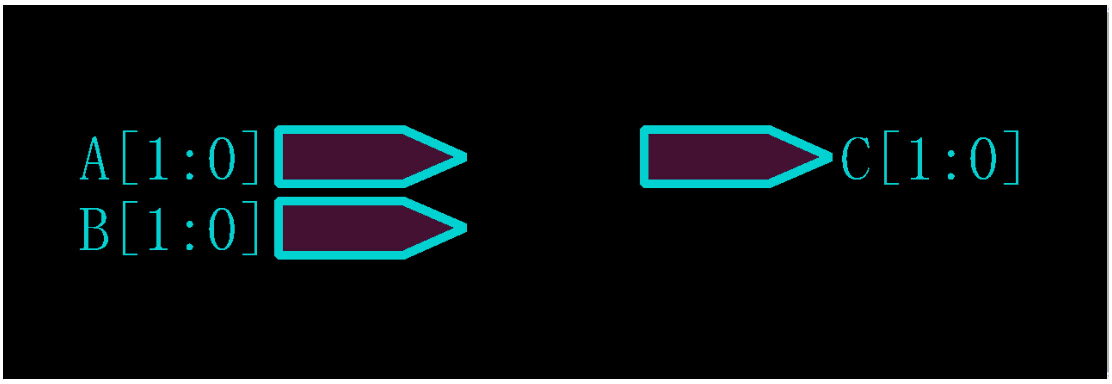


Figure 5-5 Applying the syn_black_box property to the compile netlist

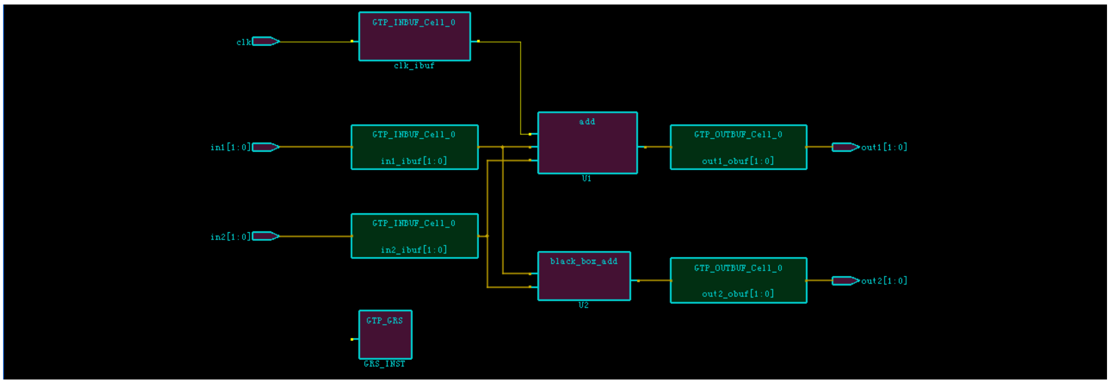


Figure 5-6 Applying the syn_black_box Attribute Synthesize Netlist

5.4 syn_direct_enable

This attribute is used to specify the enable signal for direct control of the DFF.

Pass-through inv and modules with syn_hier = "fixed/hard" set are supported.

Repeated use of syn_direct_enable in a DFF is not supported, and its use in conjunction with syn_direct_reset/ syn_direct_set in a DFF is not supported.

GLOBAL	OBJECT
no	port, net

PROPERTY VALUE	FUNCTION
0	not direct enable signal, the same Function as not setting
1	Set to direct enable signal

Syntax Specification

fdc file	`define_attribute {object} {syn_direct_enable} {value}`
verilog	`object /* synthesis syn_direct_enable = value */;`

Verilog Example

```
module direct_enable #( parameter
    size=5
    ) (
    input [size-1:0] d1,
    input clk,
    input e1,e2,
    input e3 /* synthesis syn_direct_enable = 1 */,
    output reg [size-1:0] q1
    );



    always @(posedge clk) begin
        if (e1 & e2 & e3)
```

```
                q1 = d1;
        end
    endmodule
```

As in the example before this property is used, the direct enable signal consists of e1, e2, and e3 together, as follows:

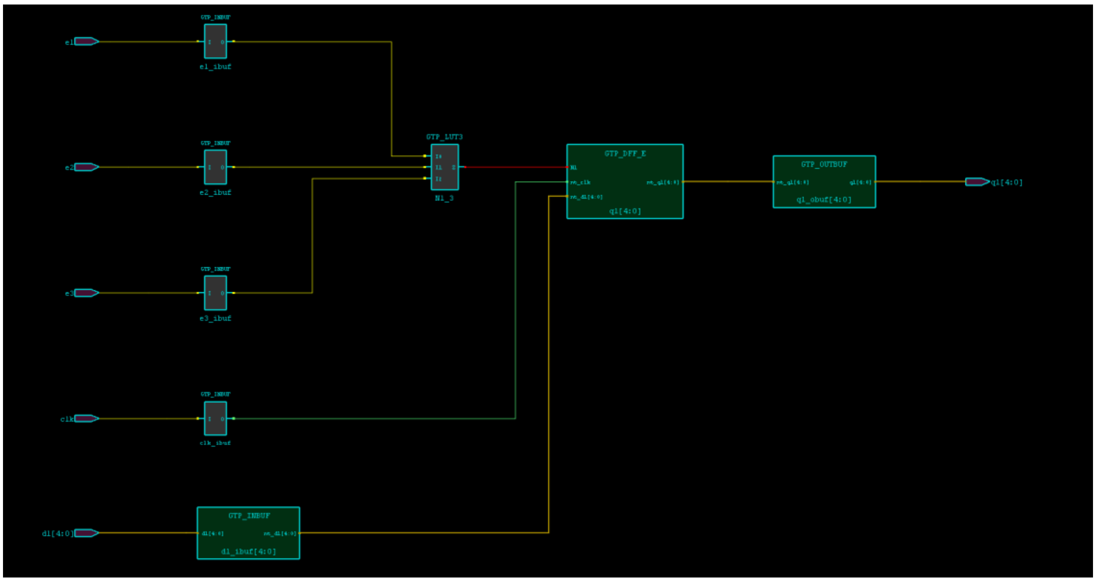


Figure 5-7 The syn_direct_enable attribute is not applied

After applying synthesis syn_direct_enable = 1, e3 acts directly as a separate enable signal, as shown below:

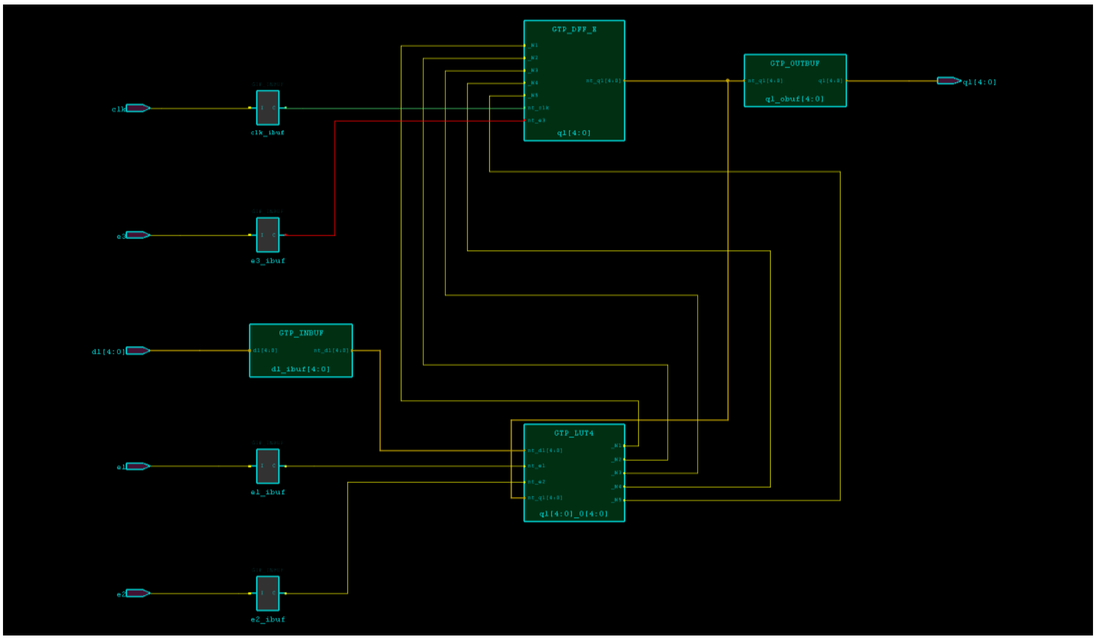


Figure 5-8 Using the syn_direct_enable Property

5.5 syn_direct_reset

This property is used to specify the reset signal that directly controls the synchronization of the DFF.

Support for pass-through inv and modules with syn_hier = "fixed/hard".

Repeated use of syn_direct_reset in a DFF and simultaneous use with syn_direct_enable/ syn_direct_set in a DFF is not supported.

syn_direct_reset Values

GLOBAL	OBJECT
no	port, net

PROPERTY VALUE	FUNCTION
0	not direct reset signal, the same Function as not setting
1	set to direct reset Signal

Syntax Specification

fdc file	define_attribute {object} { syn_direct_reset} {value}
verilog	object /* synthesis syn_direct_reset = value */;

Verilog Example

```
module direct_reset #(
    parameter size = 5
    ) (

    input a,

    input b,
    input c /* synthesis syn_direct_reset = 1 */,
    input clk,
    input [size-1:0] din,
    output reg [size-1:0] dout
    );

    always @(posedge clk) begin
        if (a == 1'b1 || b == 1'b1 || c == 1'b1)
```

```
                    dout = 0;
            else
                    dout = din;

    end

endmodule
```

As in the example before this property is used, the direct reset signal consists of a, b, and c together, as follows

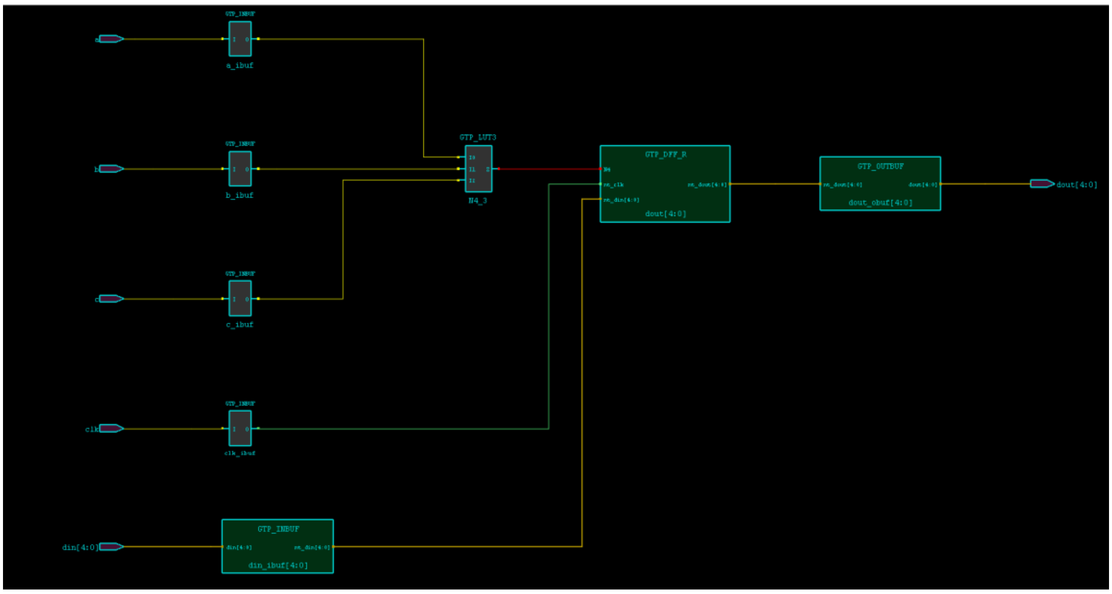


Figure 5-9 The syn_direct_reset property is not applied

With synthesis syn_direct_reset = 1 applied, c acts directly as a separate reset signal, as shown below

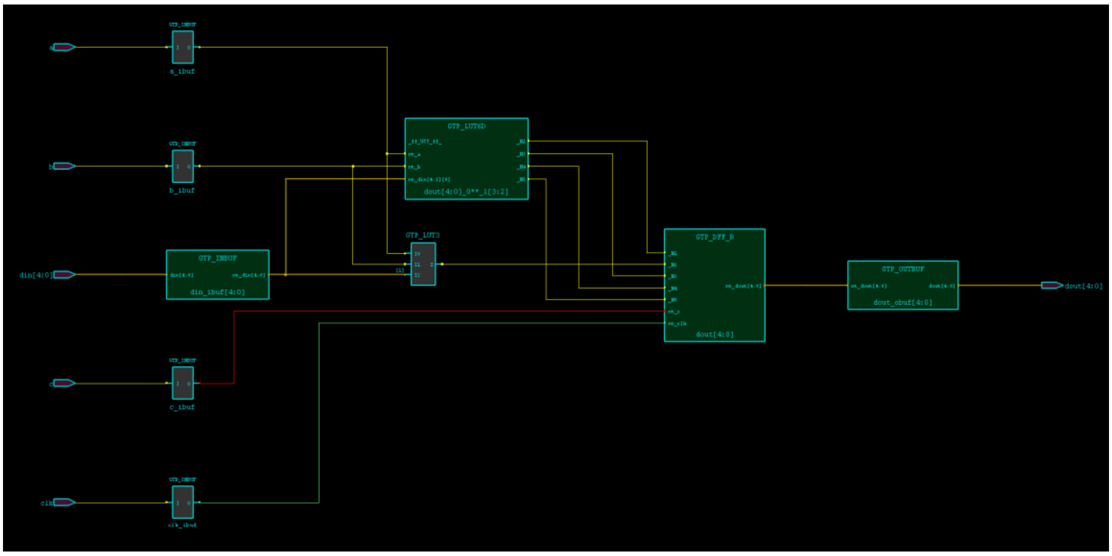


Figure 5-10 Applying the syn_direct_reset Property

5.6 syn_direct_set

This attribute is used to specify the set of signals that directly control the synchronous DFF.

Pass-through inv and modules with syn_hier = "fixed/hard" set are supported.

Repeated use of syn_direct_set in a DFF is not supported, and its use in conjunction with syn_direct_enable/ syn_direct_reset in a DFF is not supported.

syn_direct_set Values

GLOBAL	OBJECT
no	port, net

PARAMETER VALUE	FUNCTION
0	Not direct set signal, same effect as not set
1	Set to direct set signal

Syntax Specification

fdc file	define_attribute {object} { syn_direct_set} {value}
verilog	object /* synthesis syn_direct_set = value */;

Verilog Example

```
module direct_set #(
    parameter size = 5
    ) (
    input a,
    input b,
    input c /* synthesis syn_direct_set = 1
    */,
    input clk,
    input [size-1:0] din,
    output reg [size-1:0] dout
    );


    always @(posedge clk) begin
        if (a == 1'b1 || b == 1'b1 || c == 1'b1)
            dout = ~0;

        else
              dout = din;
  end

endmodule
```

As in the example before this property is used, the direct set signal consists of a, b, and c together, as follows:

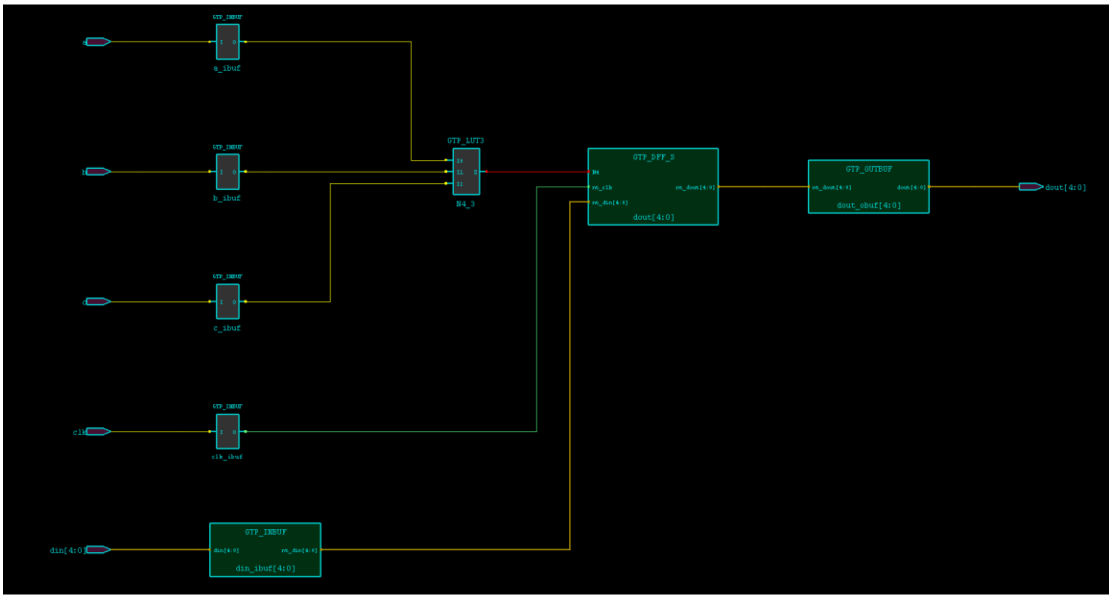


Figure 5-11 Unresponded syn_direct_set property

After applying synthesis syn_direct_set = 1, c is signaled directly as a separate set, as follows:

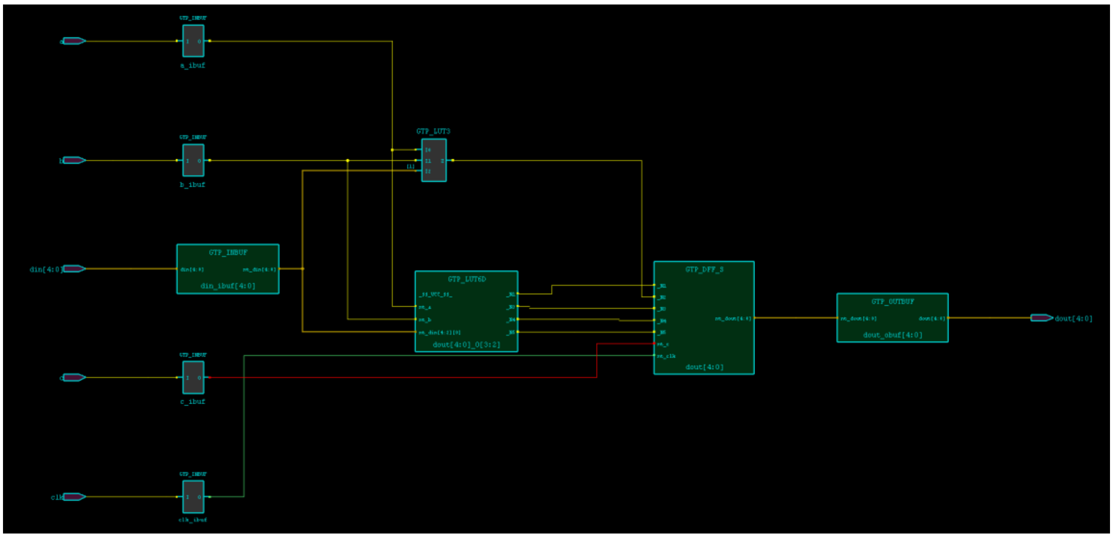


Figure 5-12 Applying the syn_direct_set Property

5.7 syn_dspstyle

Used to specify the implementation of multiplication and addition (subtraction) logic

syn_dspstyle Values

GLOBAL	OBJECT
Yes	Verilog module, operator

PARAMETER VALUE	FUNCTION
block_mult	Map multiplier, adder (subtractor) logic to APM
logic	Map multiplier logic to LUT

Syntax Specification

FDC file	`define_attribute {i: instance} syn_ dspstyle {value}`
	`Global properties:`
	`define_global_attribute    syn_ dspstyle {value}`

Verilog	`object /* synthesis syn_dspstyle = "block_mult"*/`

Verilog Example

```
module test (clk,reset,enable,a,b,q);
   input clk;
   input reset;
   input enable;
   input signed [1:0] a;
   input signed[1:0] b;
```

```
    output signed [3:0] q;

    reg signed [1:0] a_reg;
    reg signed[1:0] b_reg;
    wire signed [3:0] q1 /*synthesis syn_dspstyle = "block_mult" */;
    reg signed [3:0] q ;

    assign q1 = a_reg * b_reg;

    always @(posedge clk) begin
        if (reset)      begin
            a_reg <= 0;
            b_reg <= 0;
            q <= 0;
        end else if (enable) begin
            a_reg <= a;
            b_reg <= b;
            q <= q1 + q;
        end
    end

endmodule
```

If the example does not apply this property before, it will use ordinary logic to implement the multiplication operation.

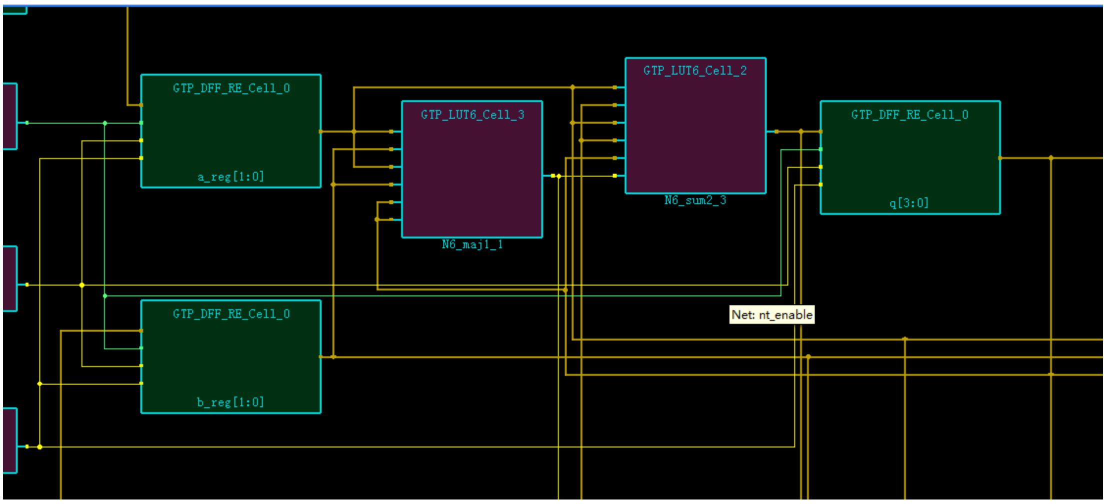


Figure 5-13 The syn_dspstyle property is not applied.

With syn_dspstyle = "block_mult" applied, multiplication is implemented using APM.

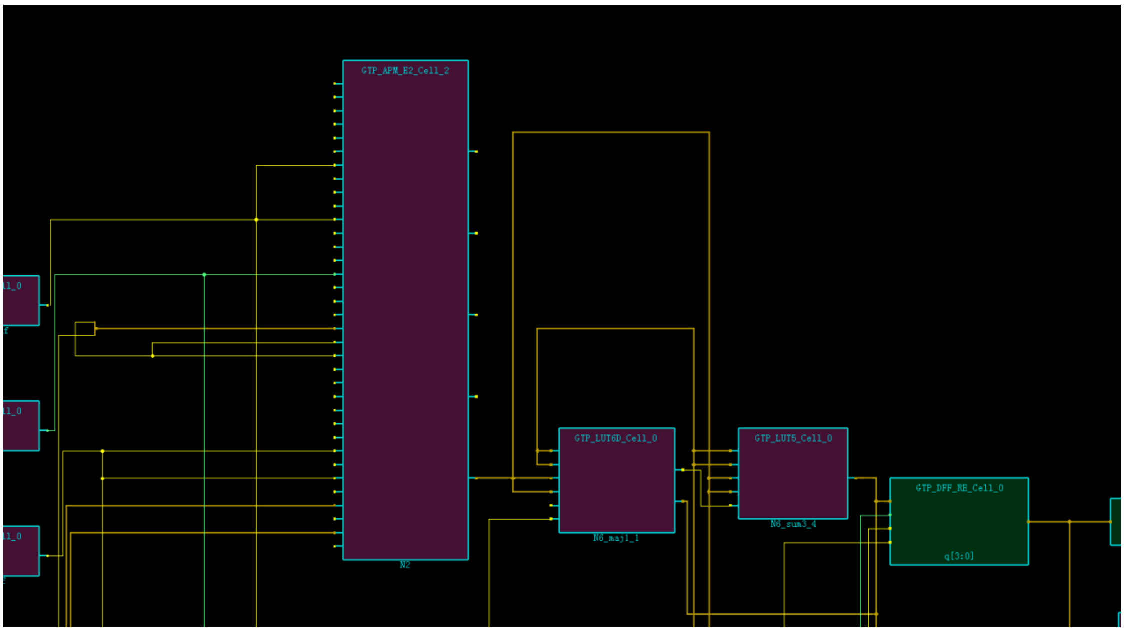


Figure 5-14 Applying the syn_dspstyle Property

5.8 syn_encoding

Used to specify the finite state machine encoding type

syn_encoding Values

GLOBAL	OBJECT
Yes	status register

PARAMETER VALUE	FUNCTION
one hot	Only one bit of each state is 1, which reduces combinational logic, is the fastest and optimizes timing, but uses more registers
gray	Only one state is flipped during adjacent state transitions, using fewer registers than onehot
sequential	Natural number encoding, using the least number of registers under the same conditions
original	Keep the original encoding in the source file
safe	Prevent the state machine from being locked in illegal state, default is onehot,safe encoding method. Once the FSM enters any illegal state, the safe logic can jump the FSM back to the legal state in the first time, which can enhance the stability of the design.
safe, one hot	Use one-hot encoding and prevent the state machine from being locked in an illegal state
safe, gray	Use Gray code encoding and prevent the state machine from being locked in an illegal state
safe,sequential	Use natural digital encoding and prevent the state machine

	from being locked in an illegal state
safe, original	Keep the original coding in the source file and prevent the state machine from being locked in an illegal state

Syntax Specification

FDC file	`define_attribute {i:instance} syn_encoding {value}`
	`Global properties:`
	`define_global_attribute syn_encoding {value}`

Verilog	`object /* synthesis syn_encoding = "onehot"*/`

Verilog Example

```
module test (clk,rst,data,out);
input clk,rst,data;
output out;
reg out;
reg [2:0] ps /* synthesis syn_encoding = "gray" */;
reg [2:0] ns;
parameter S0=3'b000, S1=3'b001, S2=3'b010, S3=3'b011, S4=3'b100,
S5=3'b101, S6=3'b110;
always@(posedge clk or posedge rst)
begin
  if(rst)
        ps <= S0;
  else
        ps <= ns;
end


always@(ps or data)
begin
  ns = S0;
  case (ps)
      S0: if(data) ns = S1;
else ns = S0;
S1: if(data)
ns = S1;
else ns = S2;
S2: if(data)
ns = S1;
else ns = S3;
S3: if(data)
ns = S4;
else ns = S0;
S4: if(data)
ns = S5;
```

```
    else ns = S2;
    S5: if(data)
    ns = S1;
    else ns = S6;
    S6: if(data)
    ns = S1;
    else ns = S3;
    default: ns = S0;
endcase
end


    always@(ps or data)
    begin
      if((ps == S6) && (data == 1))
            out = 1;
    else
    out = 0;
    end
endmodule
```

As in the example before this attribute is applied, the default encoding of the finite state machine is onehot, and the following message is typed in the run.log:

```
I: Encoding type of FSM 'ps_fsm[2:0]' is: onehot. I:
Encoding table of FSM 'ps_fsm[2:0]':
I: from   ps[2] ps[1] ps[0]
I: to   ps_6 ps_5 ps_4 ps_3 ps_2 ps_1 ps_0

I:      000     =>      0000001
I:      001     =>      0000010
I:      010     =>      0000100
I:      011     =>      0001000
I:      100     =>      0010000
I:      101     =>      0100000
I:      110     =>      1000000
```

After applying syn_encoding = "gray", the finite state machine is encoded in gray code and the following message is typed in the run.log:

```
I: Encoding type of FSM 'ps_fsm[2:0]' is: gray.
I: Encoding table of FSM 'ps_fsm[2:0]':
I: from ps[2] ps[1] ps[0]
I: to     ps_2 ps_1 ps_0
I:      000     =>      000
```

```
I:    001  =>  001
I:    010  =>  011
I:    011  =>  010
I:    100  =>  110
I:    101  =>  111
I:    110  =>  101
```

Note: The value set for the syn_encoding attribute has priority over the value set by the FSM compiler in the option.

5.9 syn_hier

The synthesis tool performs cross-level optimization by default, i.e., the synthesis process flattens the levels and then performs overall optimization. This allows for module boundaries to be crossed and for logic to be optimized across the entire design scope, which increases synthesis time but optimizes the logic, thus reducing resource consumption and increasing clock frequency.

The syn_hier attribute controls how cross-hierarchy optimization is performed..

syn_hier Values

GLOBAL	OBJECT
no	view

Parameter value	Function
fixed	By preserving the original hierarchical structure and not doing any cross-level optimization, optimizing in modules can reduce synthesis time.
hard	Maintains the original hierarchical structure, but allows for constant transfer optimization between different levels
remove	Removing the hierarchy of the current module of the tag can only remove the wrapper layer of the GTP, and there is only one GTP instance in this hierarchy (e.g., GTP_PLL_WRAPPER in the PLL IP).
macro	Retain all interfaces and contents of the current level, you must ensure that the level does not instantiate other instances, otherwise it will report an error.

Syntax Specification

FDC file	`define_attribute {v:module} syn_hier {value}`
Verilog	`object /* synthesis syn_hier = "value" */;`

Verilog Example

```
module sub(clk_sub, din_sub, dout_sub) /* synthesis syn_hier =
  "fixed" */ ;
  input clk_sub, din_sub;
  output reg dout_sub;
```

```
  always @(posedge clk_sub) begin
      dout_sub <= din_sub;
  end
endmodule

 module test(clk, dout);
  input clk;
  output dout;

   wire din;

  assign din = 1'b1;

  sub u_sub(clk, din, dout);

endmodule
```

If this property is not applied in the example, the sub in the consolidated netlist will be optimized, as shown in the figure

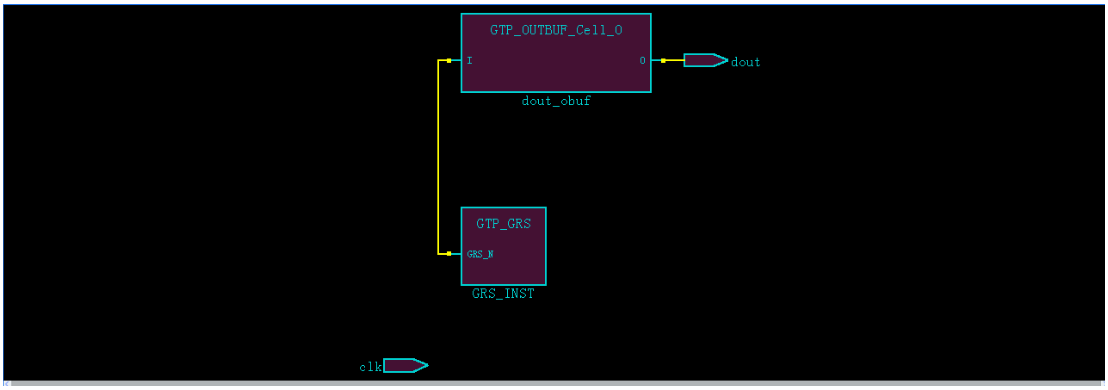


Figure 5-15 The syn_hier property is not applied

The application of syn_hier = "fixed" prevents the sub module from being optimized

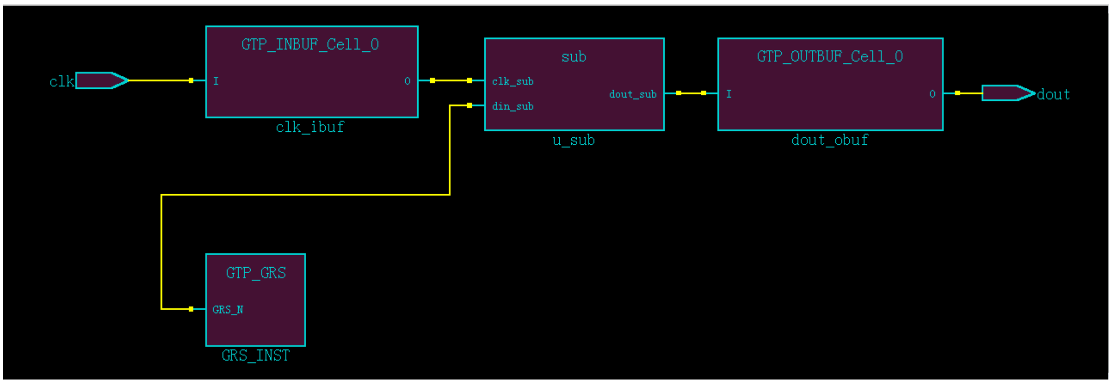


Figure 5-16 Applying the syn_hier Property

5.10 syn_insert_buffer

Used to specify the type of clock buffer used by the clock net.

syn_insert_buffer Values

GLOBAL	OBJECT
no	Verilog net, port

PARAMETER VALUE	FUNCTION
GTP_CLKBUFG	GTP_CLKBUFG after this object
GTP_CLKBUFR	GTP_CLKBUFR after this object , Compact Series does not support
GTP_CLKBUFX	GTP_CLKBUFX after this object , currently only Logos2 is supported and Titan2 series
GTP_CLKBUFM	GTP_CLKBUFX after this object , currently only Logos2 is supported and Titan2 series

Syntax Specification

Verilog	`object /* synthesis syn_insert_buffer = "GTP_CLKBUFG" */`

Verilog Example

```
module test(clk1,clk2,clk3,clk4,
       din1,din2,din3,din4,en1,en2,
       dout1,dout2,dout3,dout4);
  input clk1;
  input clk2 /* synthesis syn_insert_buffer = "GTP_CLKBUFG"
  */;
  input clk3;
  input clk4;
  input en1, en2;
  input [7:0]din1,din3;
  input din2;
  input din4;
  output [7:0]dout1,dout3;
  output dout2,dout4;
  reg [7:0]dout1,dout3;
  reg dout2,dout4;
  wire en;

  assign en = en1 & en2 ;

  always @(posedge clk1) begin
     if (en) dout1 <= din1;
```

```
        end

        always @(posedge clk2) begin
            if (en) dout2 <= din2;
        end

        always @(posedge clk3) begin
            if (en) dout3 <= din3;
        end

        always @(posedge clk4) begin
            if (en) dout4 <= din4;
        end
    endmodule
```

The sub in the consolidated netlist will be optimized if this property is not applied in the example, as follows

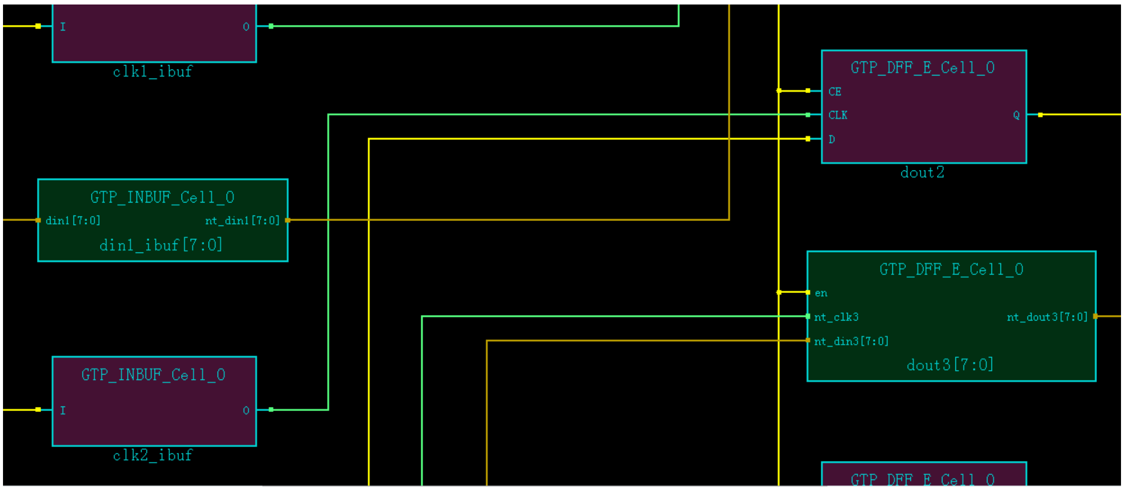


Figure 5-17 The syn_insert_buffer attribute is not applied

After applying syn_insert_buffer = "GTP_CLKBUFG", GTP_CLKBUFG will be inserted after clk2 after synthesis, as shown in the following figure

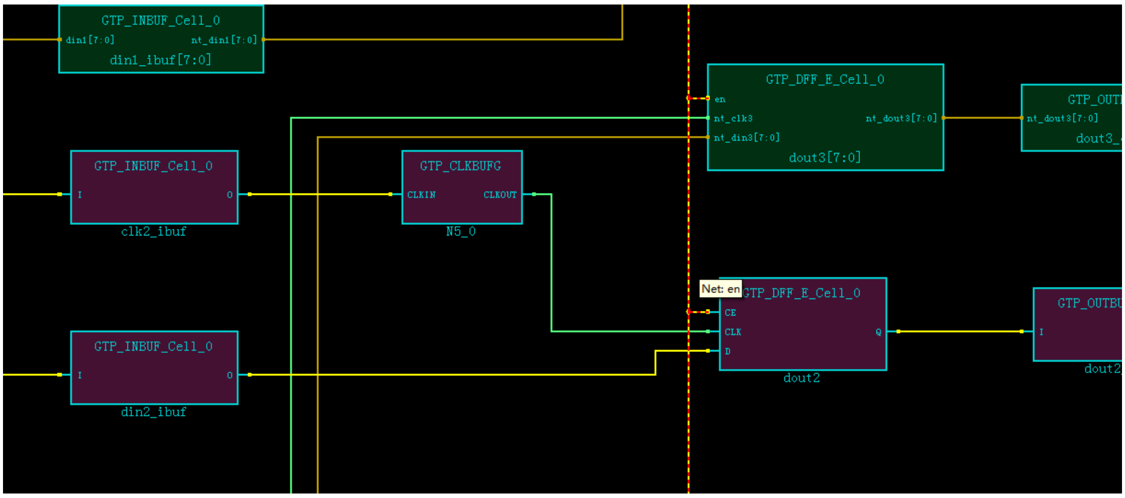


Figure 5-18 Applying the syn_insert_buffer Property

5.11 syn_keep

Specifies the net to be kept. For nets with the syn_keep attribute, the synthesis tool ensures that the net is not optimized by inserting a special keepbuf and that the original net name appears in the synthesis results file..

syn_keep Values

GLOBAL	OBJECT
no	Verilog net, port

PARAMETER VALUE	FUNCTION
1	Keep the net not optimized
0	Allow the net optimized

Syntax Specification

Verilog	`object /* synthesis syn_keep= 1 */`

Verilog Example

```
module test(out1, out2, out3, clk, a,b,c);
  output out1, out2, out3;
  input clk;
  input a,b,c;
  wire w1 /* synthesis syn_keep=1 */;
  wire w2 /* synthesis syn_keep=1 */;
  wire w3 ;
  reg out1, out2, out3;
  assign w1=a&b;
  assign w2=a&c;
  assign w3=w1&w2;
```

```
            always @(posedge clk)
            begin
               out3<=w3;
           end
    endmodule
```

As in the example, the two nets w1 and w2 in the combined net list are optimized as follows before this feature is applied

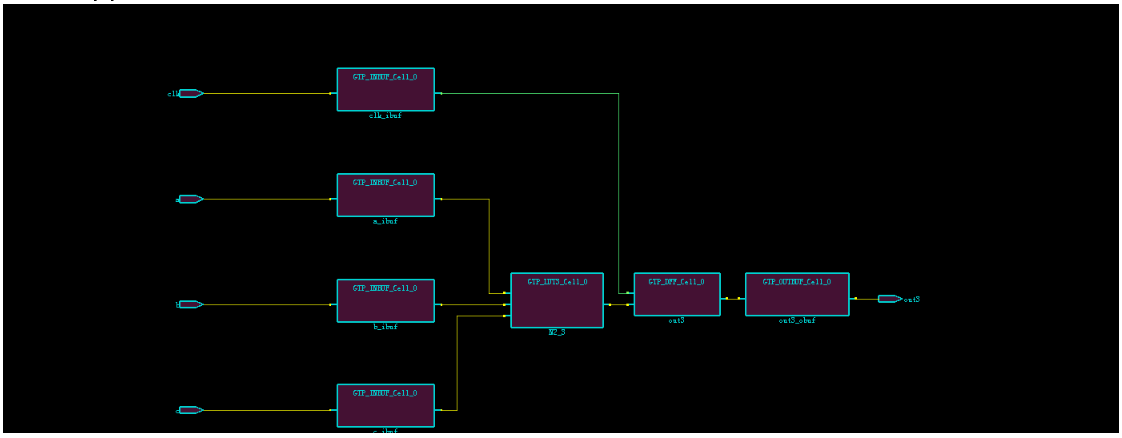


Figure 5-19 Without the syn_keep attribute applied

After applying the syn_keep=1 combination, the w1 and w2 nets in the netlist will not be optimized, as shown below

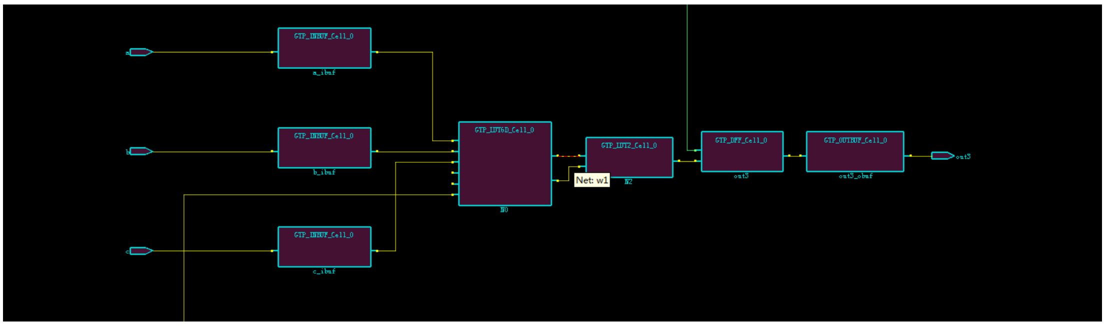


Figure 5-20 Using the syn_keep Property

Note: If the net with the syn_keep attribute set has no actual connections in the design, it will be optimized at the synthesize stage even if the keepbuf is inserted at the compile stage.

5.12 syn_looplimit

This attribute is used to specify the upper limit for the number of iterations of a Verilog loop statement (for loop, while loop, or repeat loop).

The maximum number of loop iterations in the synthesis tool is 2000. If the number of iterations exceeds two thousand, you must set the maximum number of iterations higher than the actual number of loops in the design in order to compile successfully.

syn_looplimit Values

GLOBAL	OBJECT
Yes	Verilog for-loop/while-loop/repeat- loop

PARAMETER VALUE	FUNCTION
integer	Specify the maximum number of iterations of a Verilog loop statement (for loop, while loop, or repeat loop)

Syntax Specification

verilog	`object /* synthesis syn_looplimit = value */;`

Verilog Example

```
module syn_looplimit_forloop_later (CLK, D, result);
   parameter N = 2003;
   input CLK;
   input D;
   output result;

   wire  register [0:N];

   GTP_DFF #(.INIT(1'b1))
   t1 (.CLK(CLK), .D(D), .Q(register[0]));

    genvar i;

    generate
       for(i = 0; i < N-1; i = i + 1) /*synthesis syn_looplimit = 4000
       */

       begin
          GTP_DFF #(.INIT(1'b1))
          t2 (.CLK(CLK), .D(register[i]), .Q(register[i+1]))
/*synthesis syn_noprune=1 */;
       end
```

```
        endgenerate

         assign result = register[N- 1];

    endmodule
```

If this property is not used in the example, Compile will report an error indicating that the loop is out of bounds, as shown below

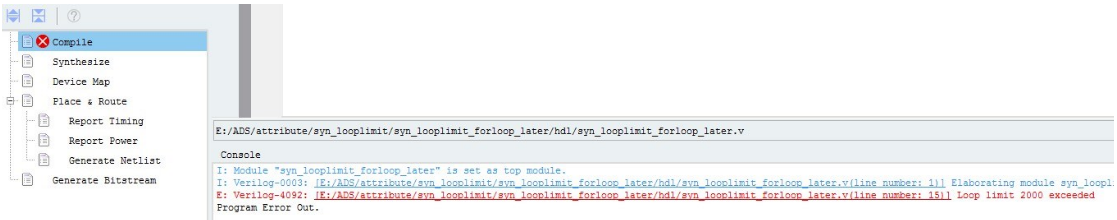


Figure 5-21 The syn_looplimit Property Is Not Applied

After applying syn_looplimit = 4000, the synthesis succeeds, as shown in the following figure

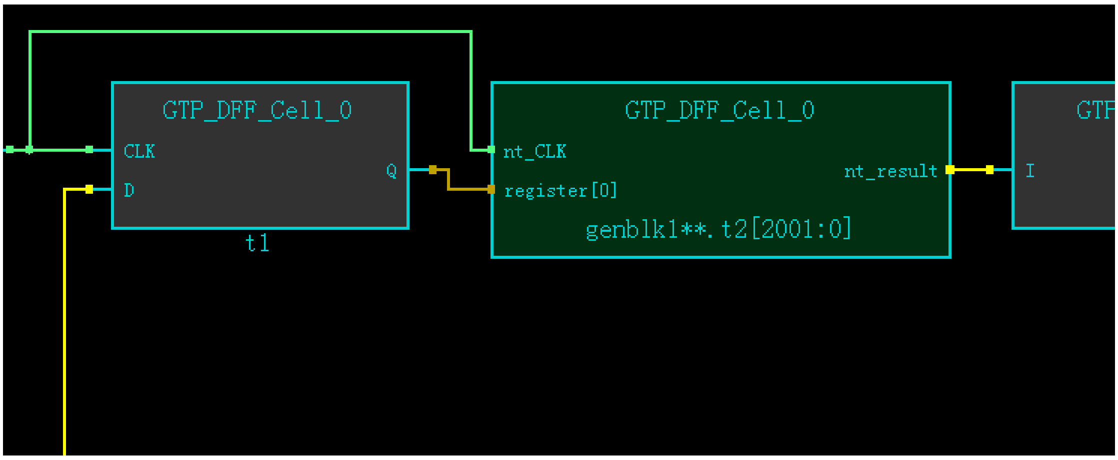


Figure 5-22 Applying the syn_looplimit Property

5.13 syn_maxfan

Sets the maximum fanout value, similar to the fanout guide in the configuration options. Use buffering to reduce the fanout of the input port, and copy logic to reduce the net driven by registers or combinational logic.

syn_maxfan Values

GLOBAL	OBJECT
Yes	module, port, net, instance, register

PARAMETER VALUE	FUNCTION
integer	Set the maximum value of fanout for the target signal

Syntax Specification

fdc file	`define_global_attribute { syn_maxfan } {value}` `define_attribute {object} { syn_maxfan } { value }`
verilog	`object /* synthesis syn_maxfan = value */;`

Verilog Example

```
module syn_maxfan_net (clk,a,b,c,q);
   input clk,a,b;
   input [7:0] c;
   output [7:0] q;


   wire en /* synthesis syn_maxfan = 4 */;
   reg [7:0] q;

   assign en = a & b;


   always @ (posedge clk) begin
   if (en)
         q = c;

   end
 endmodule
```

If this attribute is not used in the example, the target signal fanout in the integrated netlist is not restricted, as shown in the following figure

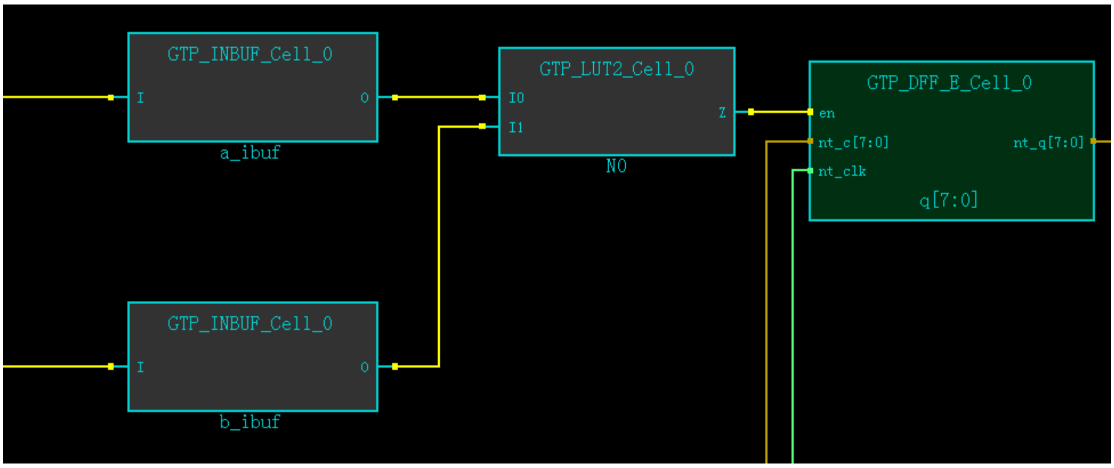


Figure 5-23 Without applying the syn_maxfan property

After applying syn_maxfan = 4, the target signal in the synthesized netlist reaches a maximum fanout of 4 by copying LUT2 as follows

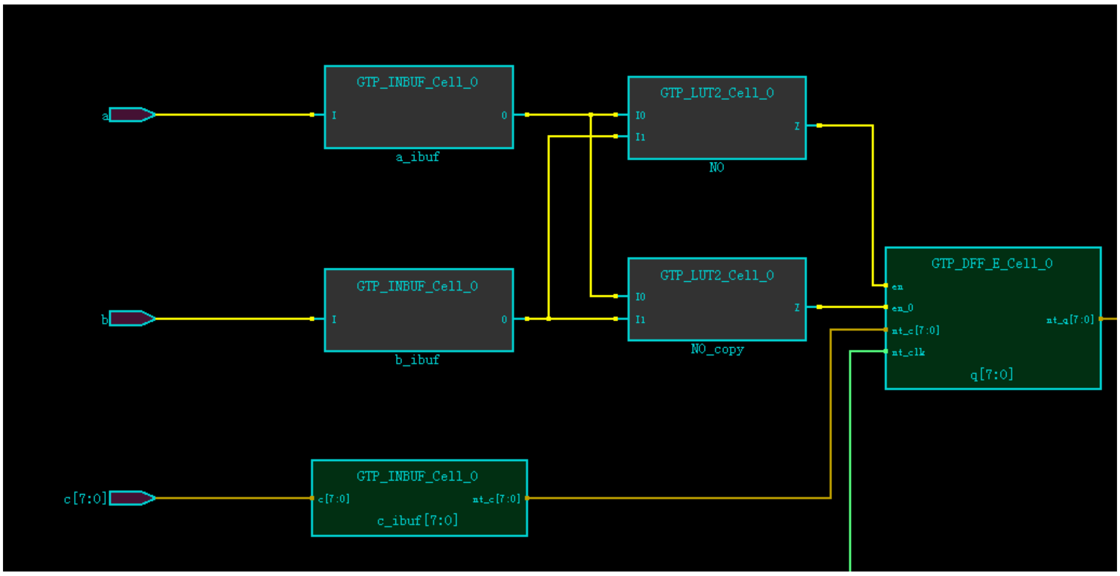


Figure 5-24 Using the syn_maxfan Property

Note: The syn_maxfan attribute setting is not valid for CLK signals and asynchronous reset signals, and only the local attribute is valid for synchronous reset signals, not the global attribute.

The value of syn_maxfan set on the attribute has higher priority than the fanout guide value in the option.

5.14 syn_noprune

This attribute is used to specify the instance that should not be deleted.

For instances with the syn_ noprune attribute, the synthesis tool does not optimize the constrained object, even if its inputs are dangling or its outputs do not drive other units.

syn_noprune Values

GLOBAL	OBJECT
no	module, instance, register

PARAMETER VALUE	FUNCTION
0	Do not keep instances that were deleted during the synthesis process
1	Retain instances that were deleted during the synthesis process

Syntax Specification

Verilog	`object /* synthesis syn_noprune = value */;`

Verilog Example

```
module syn_noprune_1_ins (input a, b, output c);
    assign c=b;
   sub i1 (a);
endmodule

module sub (input a);
   leaf i2 (a,) /* synthesis syn_noprune=1*/;
endmodule

module leaf (input a, output b) ;
   assign b = a;
endmodule
```

If this property is not used in the example, the instance(leaf) deleted during synthesis is not retained in the consolidated netlist, as shown below.

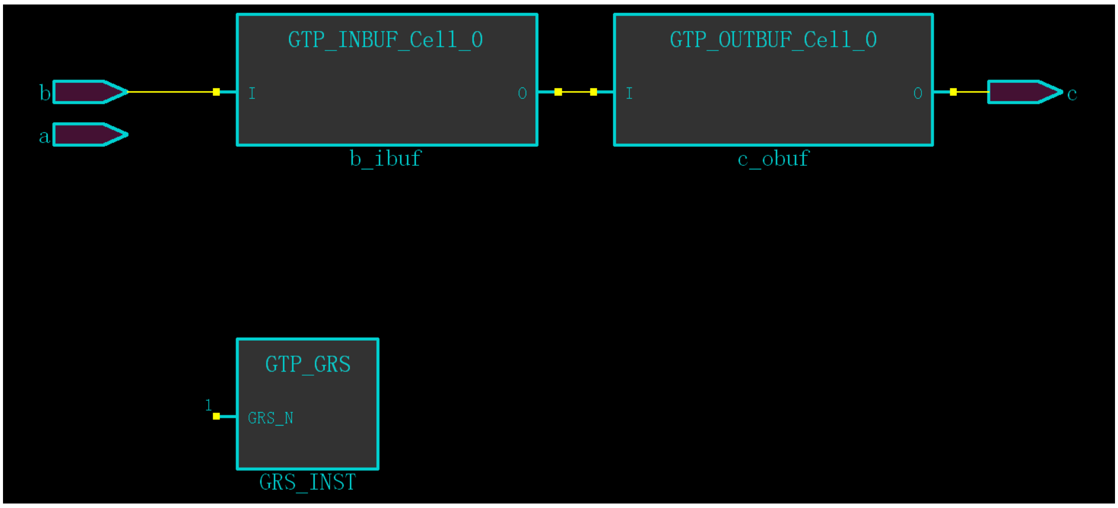


Figure 5-25 The syn_noprune attribute is not applied

After applying syn_noprune=1, the instance(leaf) that was deleted during the synthesis process will be retained in the synthesized netlist, as follows

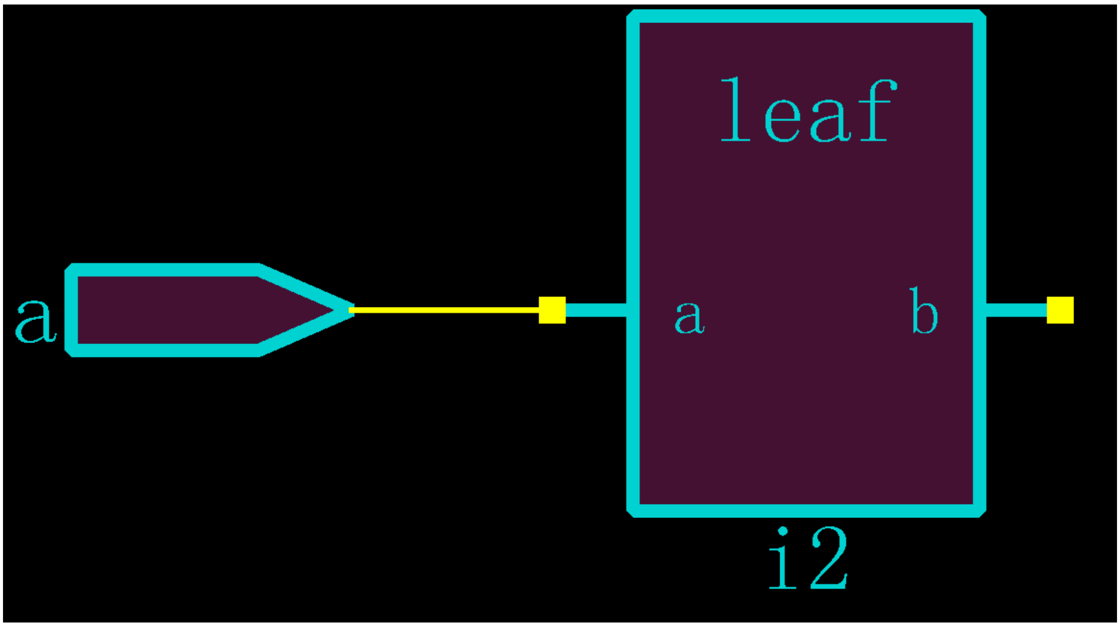


Figure 5-26 Using the syn_noprune Attribute

Note: syn_noprune can only hold inst until the synthesis phase; if the inst is not connected, it will still be optimized in dev_map.

5.15 syn_preserve

This attribute controls whether sequentials (registers) that are optimized during synthesis are preserved.

syn_preserve Values

GLOBAL	OBJECT
no	module, port, register

PARAMETER VALUE	FUNCTION
0	Registers that were optimized during the synthesis process are not retained
1	Retain registers that were optimized in the synthesis process

Syntax Specification

Verilog	`object /* synthesis syn_preserve = value */;`

Verilog Example

```
module syn_preserve_1_reg (out1,out2, data, rst, clk); output [7:0]
    out1,out2;
    input [7:0] data; input rst,clk;

    reg [7:0] out1;
    reg [7:0] out2 /* synthesis syn_preserve=1 */;
    // create the 8-bit register
    always @(posedge clk or negedge rst ) begin
        if (!rst ) begin
            out1 <= 8'b0; out2 <=
            8'b0;
        end
        else begin
            out1 <= data;
            out2 <= data;
        end
    end
endmodule
```

As in the example before using this attribute, the netlist after synthesis will not keep the register(out2[7:0]) optimized during synthesis, as follows

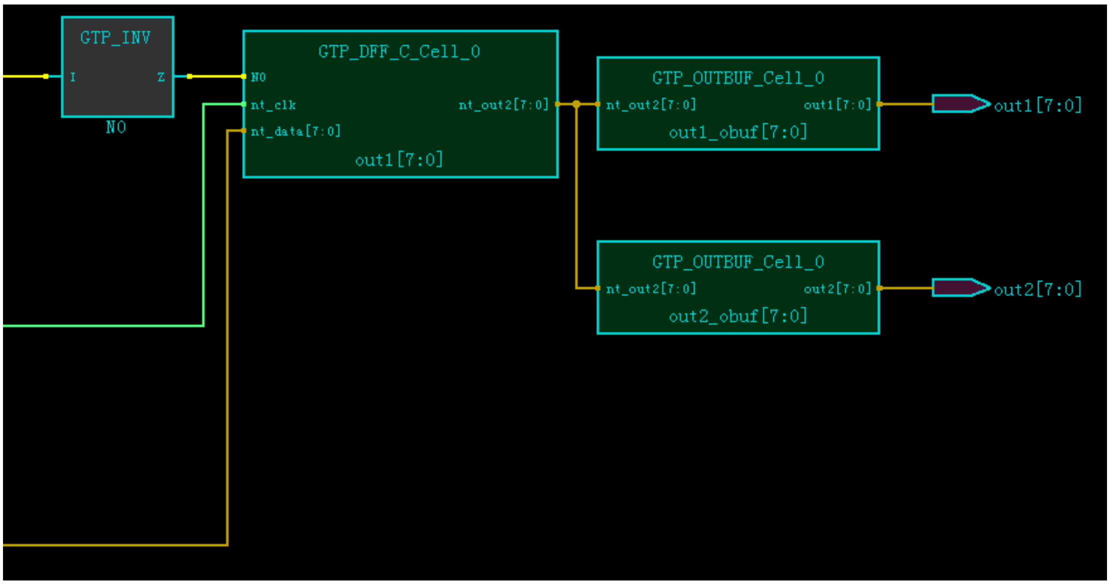


Figure 5-27 The syn_preserve attribute is not applied.

After applying syn_preserve=1, the register(out2[7:0]) optimized during the synthesis process is preserved in the synthesized netlist as follows

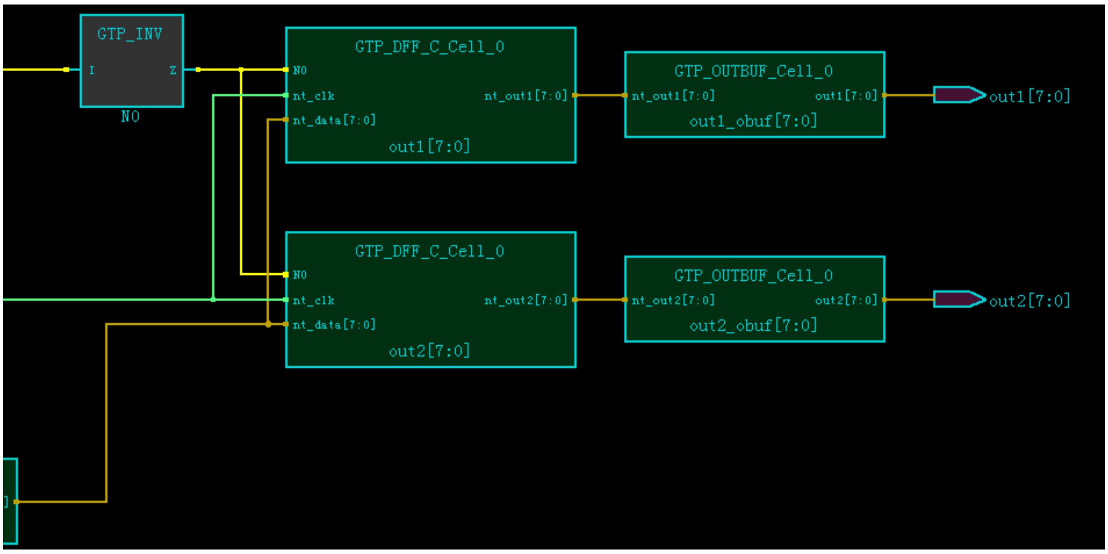


Figure 5-28 Applying the syn_preserve Property

5.16 syn_ramstyle

Used to specify how the inferred ram is implemented.

syn_ramstyle Values

GLOBAL	OBJECT
Yes	module, ram- signal

PARAMETER VALUE	FUNCTION
block_ram	Specify that inferred RAM can map to DRM resources

lut_ram	Specify that inferred RAM can map to Distributed RAM resources
registers	Specify that inferred RAM can map to Registers
no_rw_check	Infer RAM does not insert bypass logic to do read and write checks, there may be inconsistencies between RTL simulation and post-synthesis simulation
block_ram,low_power	Specify that inferred RAM can map to DRM resources, and match the data width first, then match the address.

Note: The attribute select_ram is equivalent to lut_ram

Syntax Specification

fdc file	`define_global_attribute { syn_ramstyle } {value}` `define_attribute {object} { syn_ramstyle } { value }`
verilog	`object /* synthesis syn_ramstyle ="value" */;`

Verilog Example

```
module syn_ramstyle_lut_ram_port (

    clk,we,waddr,raddr,din,q
    ) ;
    parameter ADDR_W = 8;
    parameter DATA_W = 8;

    input clk,we;
    input [ADDR_W - 1 : 0] waddr,raddr;
    input [DATA_W - 1 : 0] din;
    output [DATA_W - 1 : 0] q /* synthesis syn_ramstyle="lut_ram"
    */;

    reg [DATA_W - 1 : 0] q;
    reg [DATA_W - 1 : 0] mem [(2**ADDR_W) - 1 : 0];

    always @(posedge clk)
      if (we)
         mem[waddr] <= din;



    always @(posedge clk)
      q <= mem[raddr];

  endmodule
```

If this property is not used in the example, the DRM is inferred after synthesis, as follows

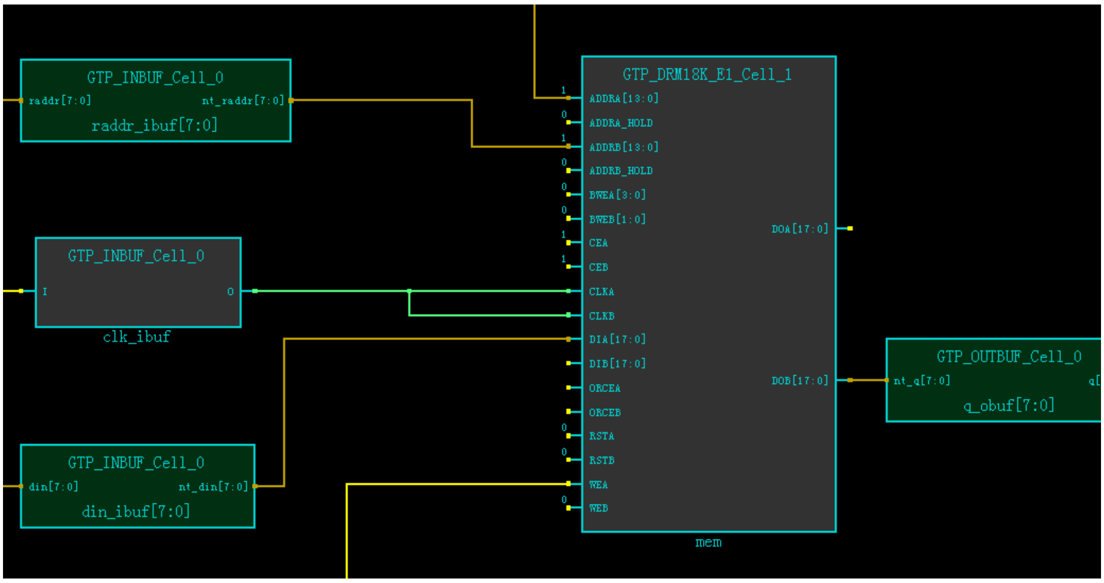
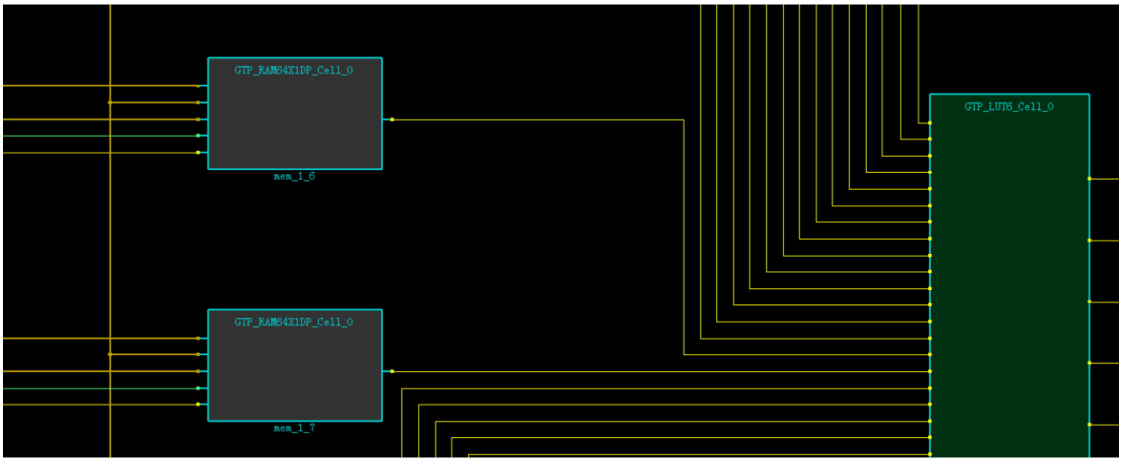


Figure 5-29 The syn_ramstyle attribute is not applied

After applying syn_ramstyle = "lut_ram", the synthesis can map to Distributed RAM, as follows


Figure 5-30 Using the syn_ramstyle Property

5.17 syn_reduce_controlset_size

This property is used to specify the minimum value of registers that share the same control port. It is equivalent to the minimal controlset size function in the configuration options.

For registers with the syn_reduce_controlset_size attribute, if the number of registers with the same control port CLK, CE, synchronous SET/RESET, etc. is less than the set value, some or all of the control pins of the corresponding FF are shifted to the D input. If the value is greater than or equal to the set value, no optimization is performed.

Control set size optimization ignores IO registers, i.e., DFFs that are connected to the port and do not have the /*synthesis syn_useioff=0*/ attribute set.

syn_reduce_controlset_size Values

GLOBAL	OBJECT
Yes	module

PARAMETER VALUE	FUNCTION
integer	If the number of registers with the same control port is less than this value, some or all of the control pins of the corresponding FF are shifted to the D input.

Syntax Specification

fdc file	`define_global_attribute { syn_reduce_controlset_size } {value}` `define_attribute {object} { syn_reduce_controlset_size } { value }`
verilog	`object /* synthesis syn_reduce_controlset_size = value */;`

Verilog Example

```
module test ( input
  clk, input en, input
  [3:0] d,
  output [3:0] q,
  output [3:0] q_e
  ) /* synthesis syn_reduce_controlset_size = 5 */;


  DFF U_DFF (
      .clk (clk),
      .d (d),
      .q (q)
      );


  DFF_E U_DFF_E (
      .clk (clk),
      .en (en),
      .d (d),
      .q_e (q_e)
      );


endmodule

module DFF (
  input clk,
  input [3:0] d,
  output [3:0] q
  );

  reg [3:0] q;

  always @ (posedge clk) begin
    q = d;
  end
```

```
endmodule


module DFF_E (
  input clk,
  input en,
  input [3:0] d,
  output [3:0] q_e
  );

  reg [3:0] q_e;

  always @ (posedge clk) begin
    if (en)
         q_e = d;

  end


endmodule
```

If this property is not used in the example, the corresponding FF will be inferred after synthesis, as follows

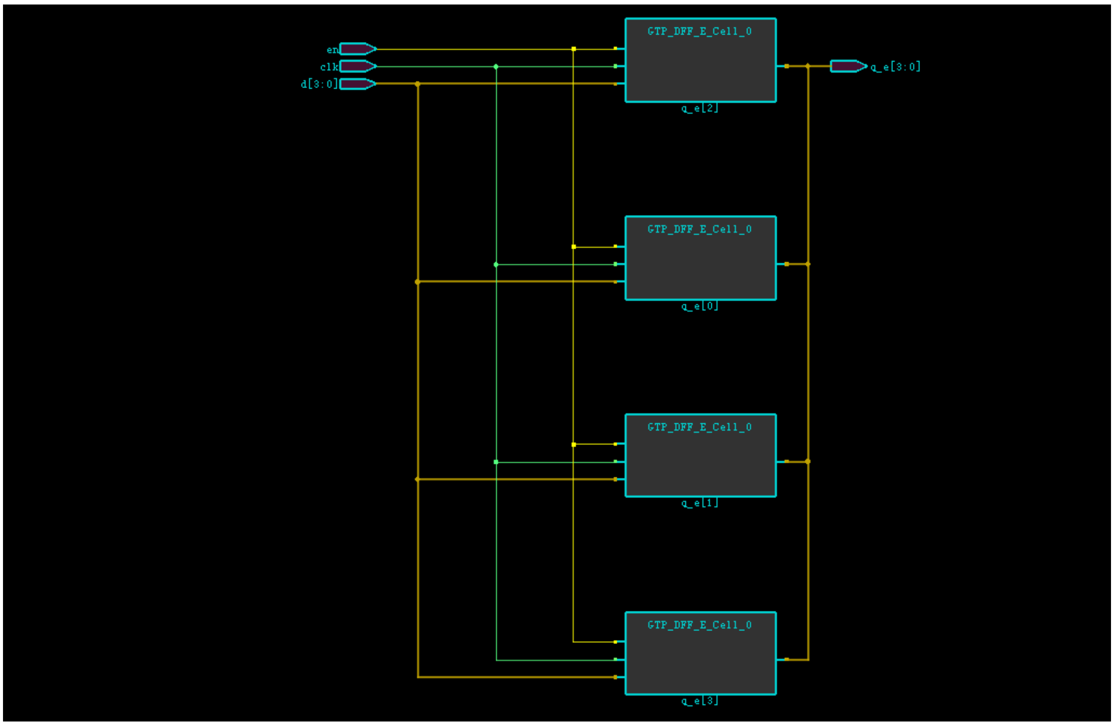


Figure 5-31 Without the syn_reduce_controlset_size property applied

After applying syn_reduce_controlset_size = 5, the synthesis will move all control pins of the corresponding FF to the D input, as follows

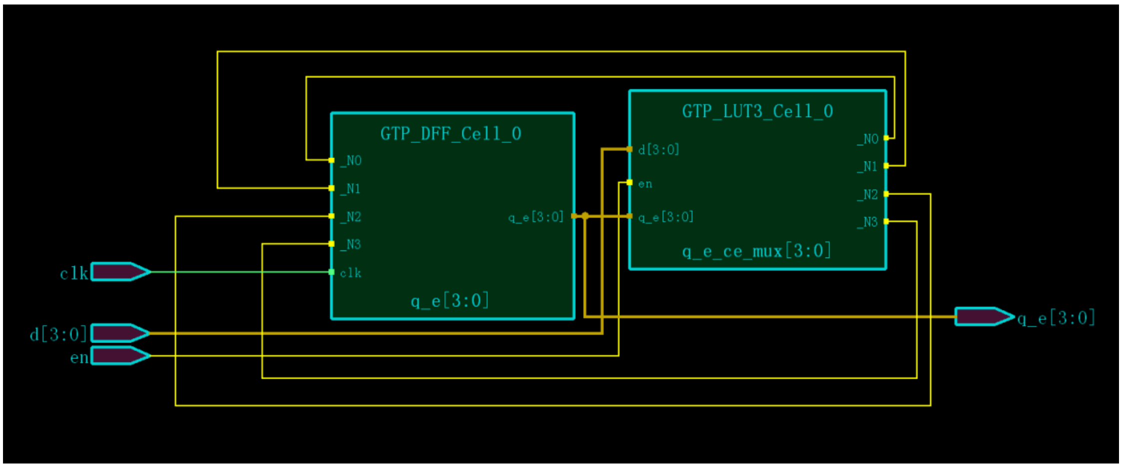


Figure 5-32 Applying the syn_reduce_controlset_size Property

5.18 syn_romstyle

Used to specify the ROM implementation style.

syn_romstyle Values

GLOBAL	OBJECT
Yes	module, rom- signal

PARAMETER VALUE	FUNCTION
block_rom	specify inferred ROM can map to DRM resource
lut_rom	specify inferred ROM can map to Distributed RAM resource
logic	specify inferred ROM can map to common logic resources

Note: The attribute select_rom is equivalent to lut_rom

Syntax Specification

fdc file	define_global_attribute { syn_romstyle } {value} define_attribute {object} { syn_romstyle } { value }
verilog	object /* synthesis syn_romstyle ="value" */;

Verilog Example

```
module syn_romstyle_block_rom_reg (clk,addr,dout) ;
  input clk;
  input [4:0] addr;
  output [7:0] dout;
  reg [7:0] dout /* synthesis syn_romstyle = "block_rom"*/;
  reg [4:0] addr_reg;

  always @(posedge clk) begin
     addr_reg<=addr;
```

```
        case (addr_reg)
            5'b00000: dout <= 8'b10000011;
            5'b00001: dout <= 8'b00000101;
            5'b00010: dout <= 8'b00001001;
            5'b00011: dout <= 8'b00001101;
            5'b00100: dout <= 8'b00010001;
            5'b00101: dout <= 8'b00011001;
            5'b00110: dout <= 8'b00100001;
            5'b00111: dout <= 8'b10110100;
            5'b01000: dout <= 8'b11000000;
            5'b01000: dout <= 8'b00011011;
            5'b01001: dout <= 8'b10110001;
            5'b01010: dout <= 8'b00110101;
            5'b01011: dout <= 8'b01110010;
            5'b01100: dout <= 8'b11100011;
            5'b01101: dout <= 8'b00111111;
            5'b01110: dout <= 8'b01010101;
            5'b01111: dout <= 8'b00110100;
            5'b10000: dout <= 8'b10110000;
            5'b10000: dout <= 8'b11111011;
            5'b10001: dout <= 8'b00010001;
            5'b10010: dout <= 8'b10110011;
            5'b10011: dout <= 8'b00101011;
            5'b10100: dout <= 8'b11101110;
            5'b10101: dout <= 8'b01110111;
            5'b10110: dout <= 8'b01110101;
            5'b10111: dout <= 8'b01000011;
            5'b11000: doubt <= 8'b01011100;
            5'b11000: doubt <= 8'b11101011;
            5'b11001: doubt <= 8'b00010100;
            5'b11010: doubt <= 8'b00110011;
            5'b11011: doubt <= 8'b00100101;
            5'b11100: dout <= 8'b01001110;
            5'b11101: dout <= 8'b01110100;
            5'b11110: dout <= 8'b11100101;
            default: dout <= 8'b01111110;
       endcase
    end

endmodule
```

If the example does not use this property before, after the synthesis infer out ROM, as follows

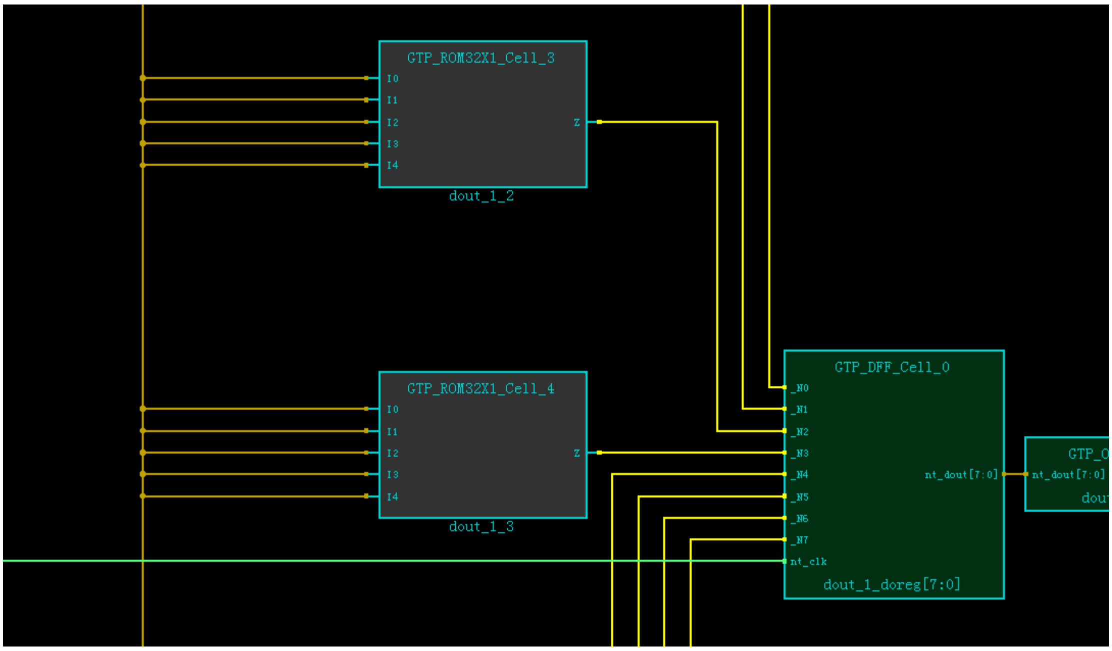


Figure 5-33 The syn_romstyle attribute is not applied.

After applying syn_romstyle = "block_rom", the DRM can be inferred after synthesis, as follows

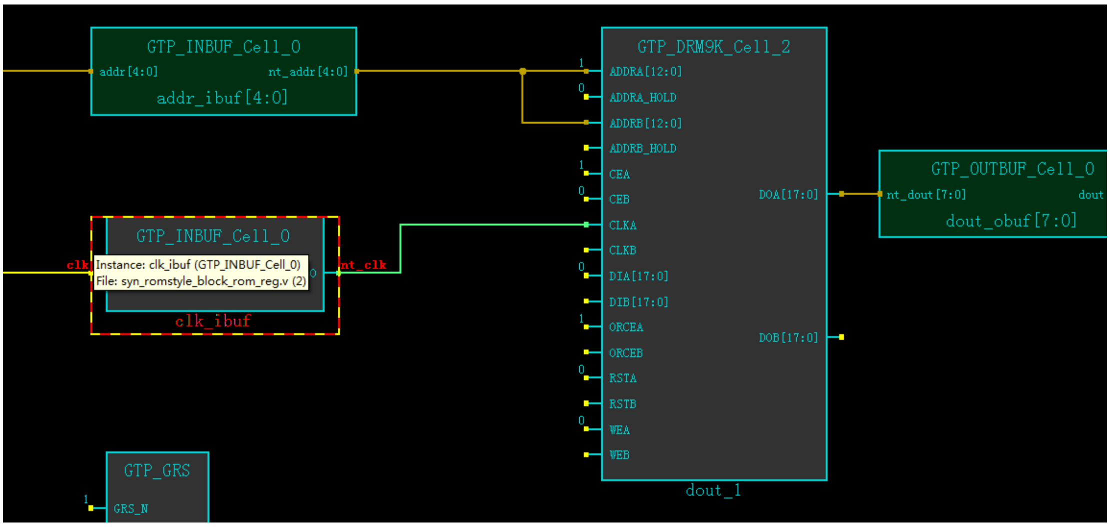


Figure 5-34 Applying the syn_romstyle Property

5.19 syn_srlstyle

This attribute is used to specify how the inferred shift register is implemented.

syn_srlstyle Values

GLOBAL	OBJECT
Yes	module, instance, register

PARAMETER VALUE	FUNCTION
block_ram	specify infered shift register Can map to DRM resource
lut_ram	specify infered shift register Can map to Distributed RAM resource
registers	specify infered shift register Can map to Registers

Note: The attribute select_ram is equivalent to lut_ram

Syntax Specification

fdc file	define_global_attribute { syn_srlstyle } {value} define_attribute {object} { syn_srlstyle } { value }
verilog	object /* synthesis syn_srlstyle ="value" */;

Verilog Example

```
module test(
   input din,
   input en,
   output dout1,
   input clk
   );

   reg [3:0] pool /* synthesis syn_srlstyle = "block_ram" */;

   always @(posedge clk) begin
      if (en)

         pool <= {pool[2:0], din};

   end

   assign dout1 = pool[3];

endmodule
```

If the example does not use this property before, after the synthesis infer out Registers, the following figure

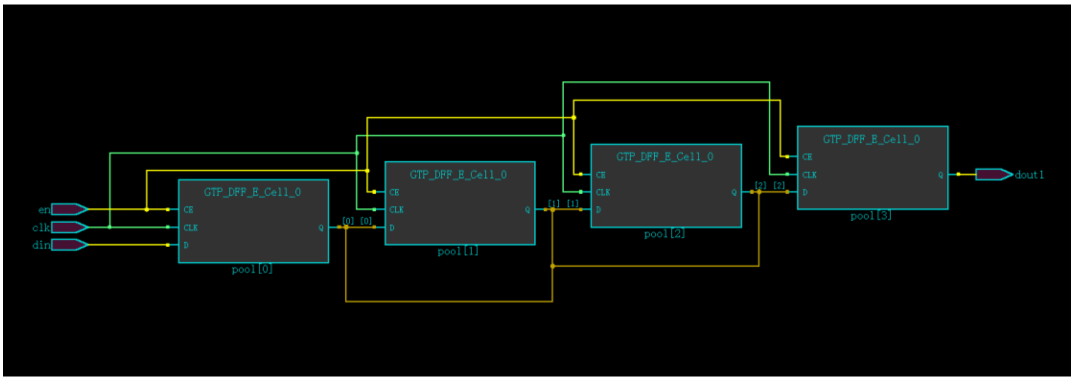


Figure 5-35 The syn_srlstyle attribute is not applied

After applying syn_srlstyle = "block_ram", the composite can be mapped to DRM as follows

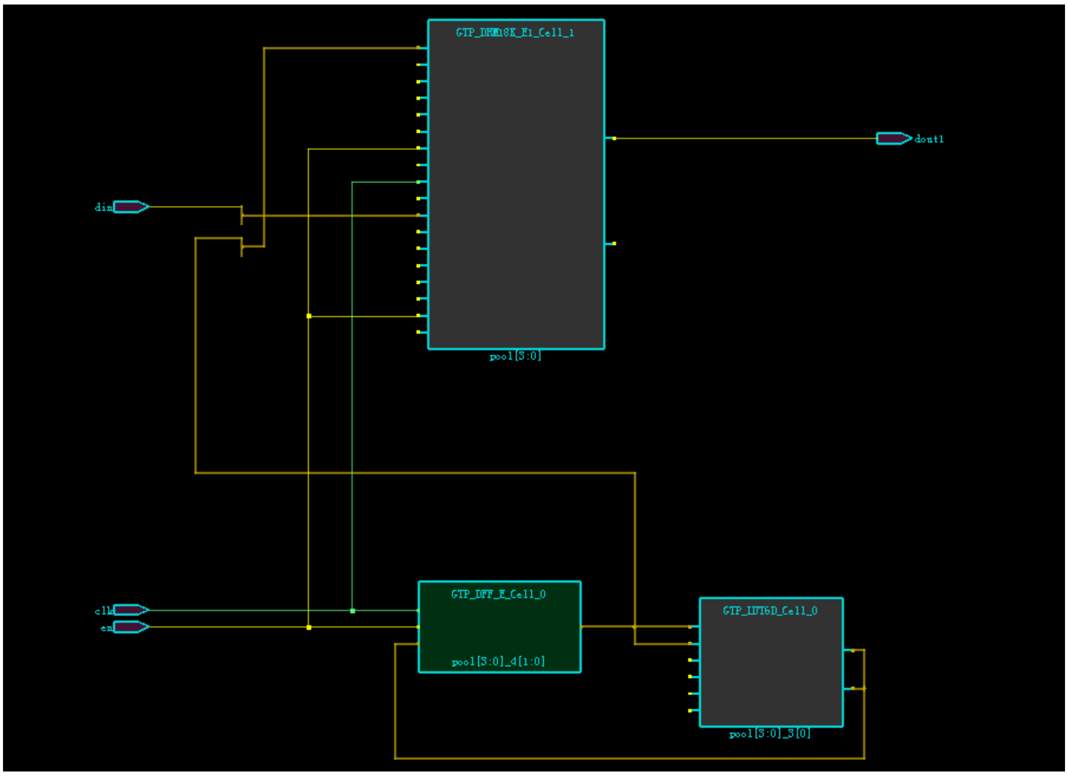


Figure 5-36 Using the syn_srlstyle property

5.20 syn_state_machine

Specifies whether the register is an infer FSM.

Note: 1. If the attribute is set on a module, it is only supported to be set on the module of the current state machine implementation code.

2. If the value of the syn_state_machine attribute on the register is not set, the fsm_compiler configuration option determines whether to infer FSM or not. If syn_preserve is also set, the syn_preserve attribute has the highest priority; syn_preserve=1, no infer fsm; syn_preserve=0, if the value of the syn_state_machine attribute on the register is 1 (the value of the attribute on the register overrides the value of the attribute on the module), FSM is inferred.), FSM is inferred; if the value of the syn_state_machine attribute of the register is 0, FSM is not inferred.

syn_state_machine Values

GLOBAL	OBJECT
no	module, register

PARAMETER VALUE	FUNCTION
1	Compile phase inferring finite state machines
0	The Compile phase does not infer a finite state machine

Syntax Specification

Verilog	`object /* synthesis syn_state_machine= 1*/`

Verilog Example

```
module test (clk,rst,data,out);
input clk,rst,data;
output out;
reg out;
reg [2:0] ps /* synthesis syn_state_machine = 0 */;
reg [2:0] ns;
parameter S0=3'b000, S1=3'b001, S2=3'b010, S3=3'b011,
S4=3'b100, S5=3'b101, S6=3'b110;

always@(posedge clk or posedge rst)
begin
      if(rst)
            ps <= S0;
      else
      ps <= ns; end

always@(ps or data)
begin
      ns = S0;
            case (ps)
                  S0: if(data)
                  ns = S1;
            else ns = S0;
                  S1: if(data)
                  ns = S1;
            else ns = S2;
                  S2: if(data)
                  ns = S1;
            else ns = S3;
                  S3: if(data)
                  ns = S4;
```

```
                else ns = S0;
                      S4: if(data)
                      ns = S5;
                else ns = S2;
                      S5: if(data)
                      ns = S1;
                else ns = S6;
                      S6: if(data)
                      ns = S1;
                else ns = S3;
                default: ns = S0;
          endcase
    end

    always@(ps or data)
    begin
          if((ps == S6) && (data == 1))
                out = 1;
          else
                out = 0;
    end
endmodule
```

If this attribute is not applied in the example, the default is to infer the finite state machine with the encoding type of onehot and to write the following message to run.log

I: Encoding type of FSM 'ps_fsm[2:0]' is: onehot.

When syn_state_machine = 0 is used, the finite state machine is not inferred.

5.21 syn_unconnected_inputs

This attribute is used to define that the unconnected inputs of the instance remain suspended in the synthesis result table.

Normally, the ADS synthesis tool ensures that the instance pin has a connected net and the net has a driver, but the user can use the syn_unconnected_inputs attribute to define exceptional pins, such as hardwired pins in a GTP, with this attribute set.

When set to a module, all unconnected inputs for the module instance are left empty in the synthesis result net list. When set to an instance, the unconnected inputs of that instance are left unconnected in the consolidated results netlist.

Note: An unconnected input means that the input pin is not connected to a signal trace. If the input pin is connected to a net but the net has no driver, the unconnected net is optimized to GND in the synthesis results netlist.

syn_unconnected_inputs Values

GLOBAL	OBJECT
no	module, instance

PARAMETER VALUE	FUNCTION
@all	For all unconnected inputs the combined results netlist will remain empty
<PIN_NAME>	Only the target unconnected inputs will remain suspended in the combined results netlist

Syntax Specification

fdc file	`define_attribute {object} { syn_unconnected_inputs } { value }`
verilog	`object /* synthesis syn_unconnected_inputs ="value" */;`

Verilog Example

```
module USER_I2C (
    output      SCL_OE_N,
    input       SCL_I,
    output      SCL_O,
    output      SDA_OE_N,
    input       SDA_I,
    output      SDA_O,
    output      IRQ
) /* synthesis syn_black_box syn_unconnected_inputs="SCL_I,SDA_I" */;


endmodule


module syn_unconnected_inputs_pins (
    output      SCL_OE_N,
    input       SCL_I,
    output      SCL_O,
    output      SDA_OE_N,
    input       SDA_I,
    output      SDA_O,
    output      IRQ

);


USER_I2C T_USER_I2C    (
    .SCL_OE_N(SCL_OE_N),
    .SCL_I(),
    .SCL_O(SCL_O),
    .SDA_OE_N(SDA_OE_N),
    .SDA_I(),
    .SDA_O(SDA_O),
    .IRQ(IRQ)
```

```
        );

endmodule
```

If this attribute is not used in the example, the PIN "SCL_I" and PIN " SDA_I" will be grounded in the integrated netlist, as follows

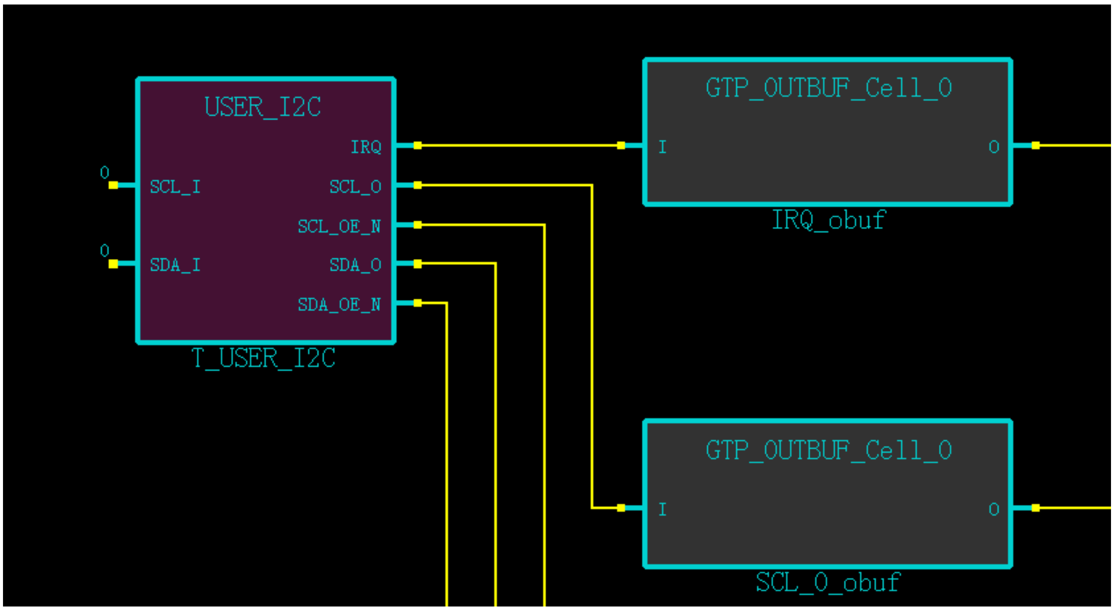


Figure 5-37 The syn_unconnected_inputs Attribute Is Not Applied

After applying syn_unconnected_inputs="SCL_I,SDA_I", PIN "SCL_I" and PIN "SDA_I" are left empty in the synthesized netlist, as shown in the following figure.

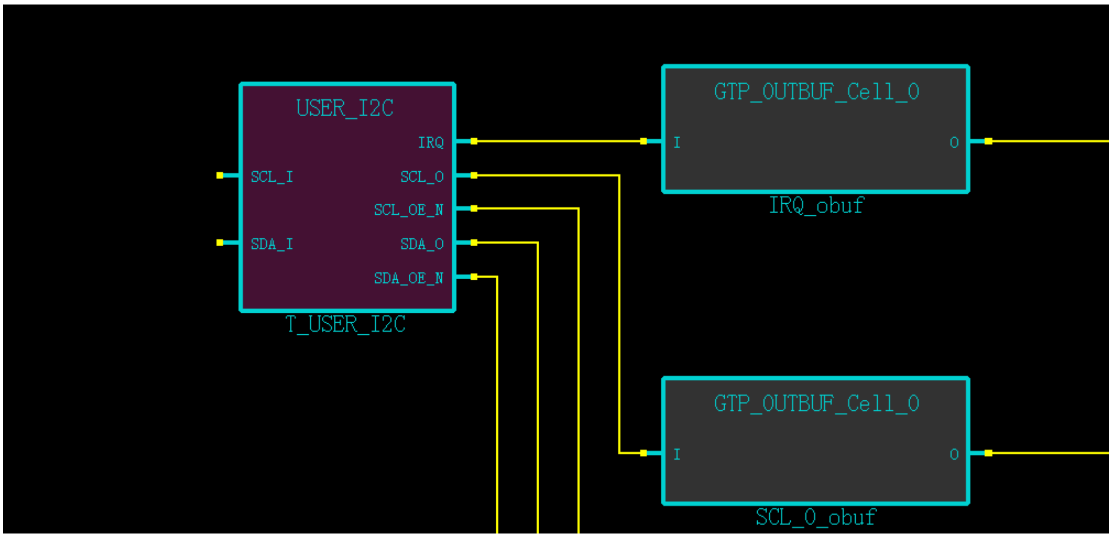


Figure 5-38 Applying the syn_unconnected_inputs Property

5.22 syn_useioff

This attribute is used to specify whether FF is packed with IO. It can be set locally in a specific port and register, or globally in a module.

For register or port with syn_useioff =1 attribute, the synthesized object will have PAP_IO_REGISTER

attribute on it, thus packing IO and FF together in the Device Map phase.

If syn_useioff = 0, the tagged IO and REGISTER are not packed, and if set to register, the register must be connected to the port for this attribute to take effect.

syn_useioff Values

GLOBAL	OBJECT
Yes	port, register

PARAMETER VALUE	FUNCTION
0	Does not pack FF with IO
1	Will pack FF with IO

Syntax Specification

fdc file	`define_global_attribute {syn_useioff} {value}` `define_attribute {object} {syn_useioff} { value }`
verilog	`object /* synthesis syn_useioff = value */;`

Verilog Example

```
module syn_useioff_1_reg (
    input clk,
    input d,
    output q
);


    test U (
        .clk(clk),
        .d(d),
        .q(q)
         );
     endmodule

module test (
    input clk, input
    d, output q
);

    reg temp;
    reg qreg /* synthesis syn_useioff=1 */;

    assign q = qreg;
```

```
always@(posedge clk) begin temp
    <= d;
    qreg <= temp;

end


endmodule
```

If this property is not used in the example, the Device Map phase will not pack q and qreg together, as shown below

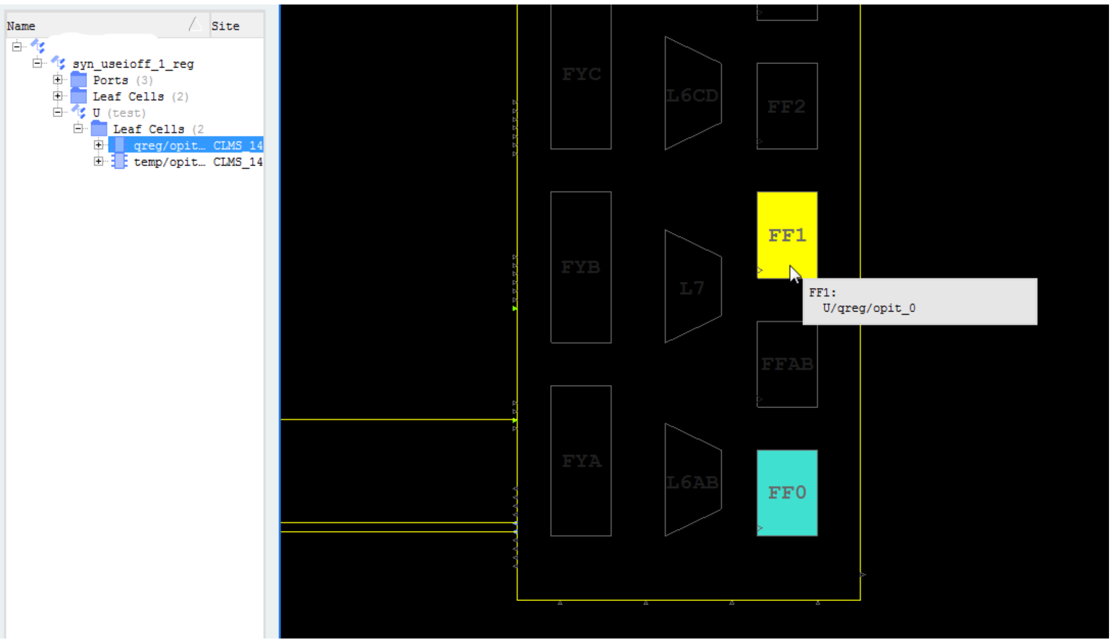


Figure 5-39 The syn_useioff attribute is not applied

After applying syn_useioff=1, the Device Map stage will pack q and qreg together, as follows

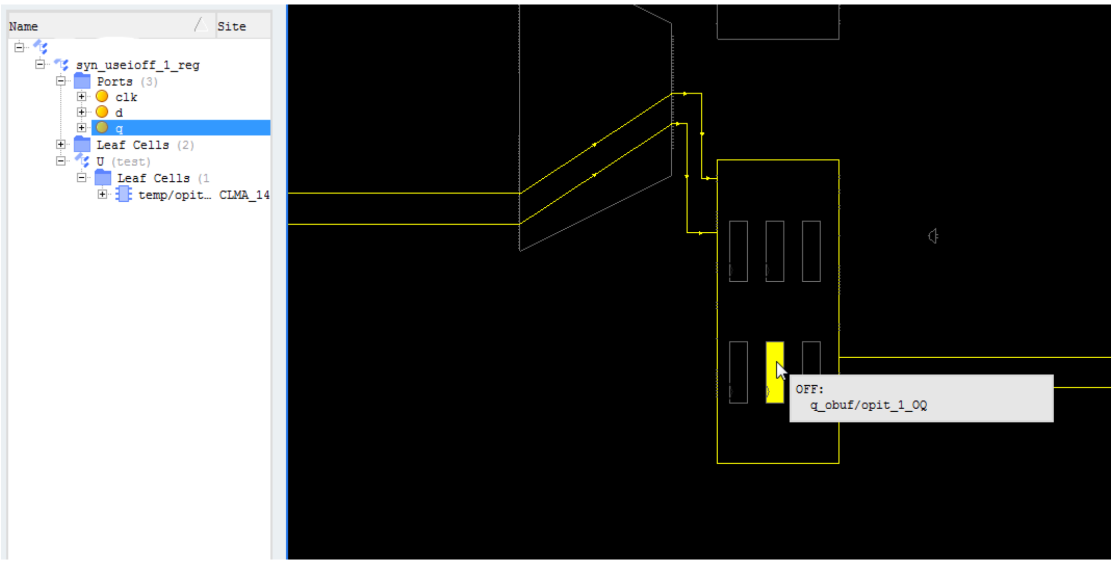


Figure 5-40 Applying the syn_useioff Property

5.23 full_case

Indicates that all possible values have been given in the case statement and that the synthesizer does not need to infer out the latch to retain the value.

Notice:

full_case Values

GLOBAL	OBJECT
no	A case, case, or casez statement declaration

Syntax Specification

Verilog	object /* synthesis full_case */

Verilog Example

```
module test (out, a, b, c, d, select);
      output out;
      input a, b, c, d;
      input [3:0] select;
      reg out;

always @(select or a or b or c or d)
begin
      casez (select) /* synthesis full_case */
      4'b0001: out = a;
      4'b001?: out = b;
      4'b01??: out = c;
      4'b1???: out = d;
      endcase
end
endmodule
```

If this property is not used in the example, the composite will infer latch, as follows

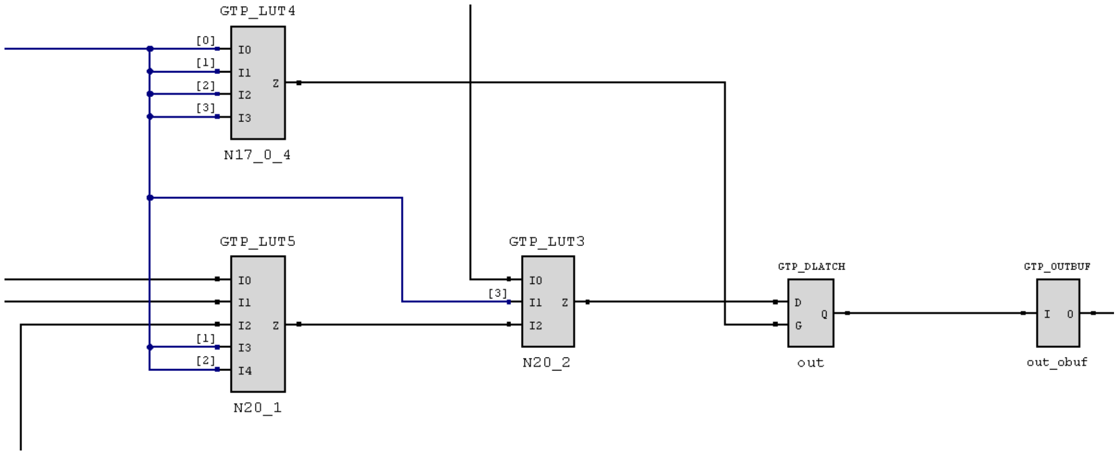


Figure 5-41 Not Applying the full_case Property

When the full_case property is applied, the latch is not inferred.

Note: Applying full_case may result in inconsistencies between the rtl simulation and the synthesized simulation.

5.24 parallel_case

Applies only to Verilog designs and forces the use of parallel multiplexing structures instead of priority coding structures.

parallel_case Values

GLOBAL	OBJECT
no	A case,casex, or casez statement declaration

Syntax Specification

Verilog	object /* synthesis parallel_case */

Verilog Example

```
module test (out, a, b, c, d, select);
      output out;
      input a, b, c, d;
      input [3:0] select;
      reg out;

      always @(select or a or b or c or d)
      begin
            casez (select) /* synthesis parallel_case */
                  4'b???1: out = a;
                  4'b??1?: out = b;
                  4'b?1??: out = c;
                  4'b1???: out = d;
```

```
            endcase
        end
    endmodule
```

If this property is not used in the example, the priority coding structure is inferred, as follows

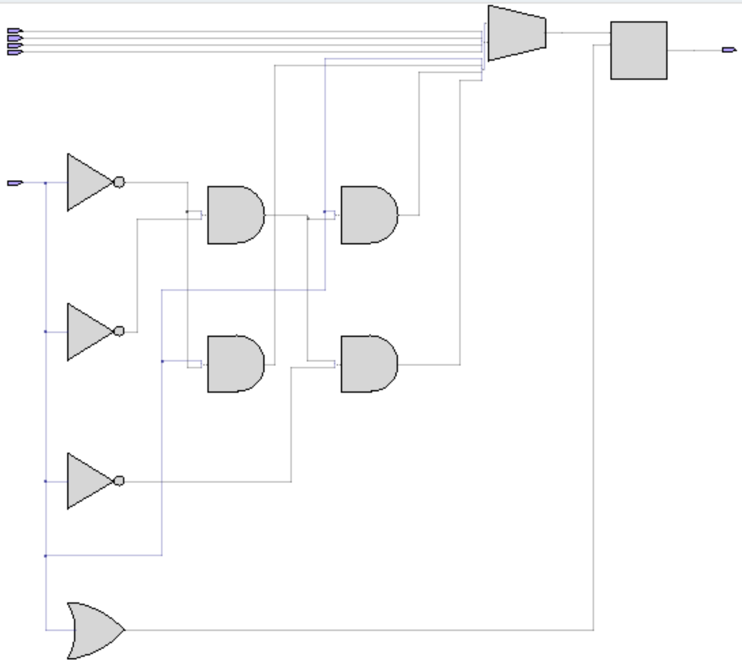


Figure 5-42 The parallel_case attribute is not applied.

After the parallel_case attribute is applied, the parallel multiplexing structure is inferred.

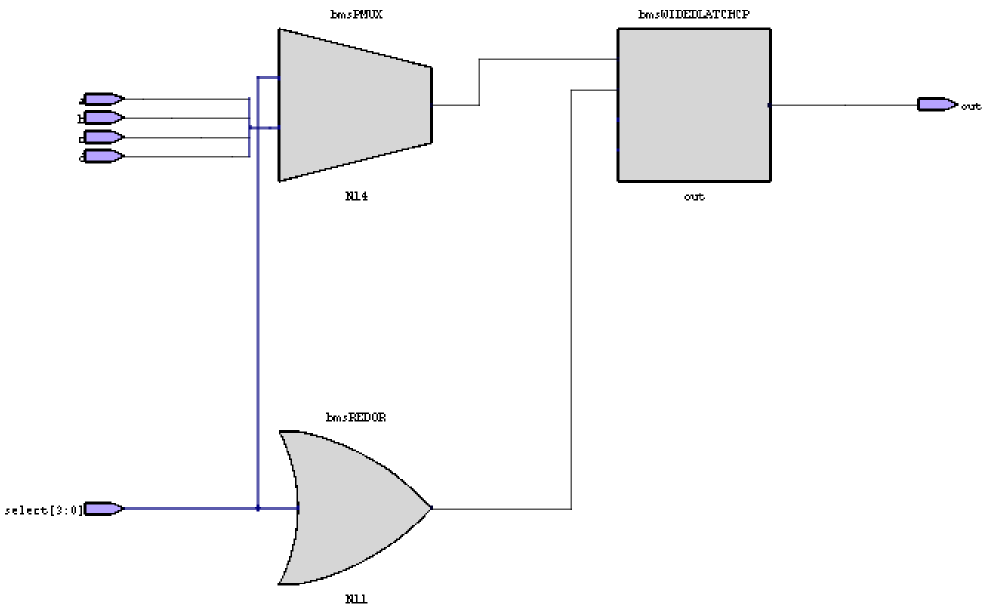


Figure 5-43 Applying the parallel_case property

5.25 translate_off/ translate_on

Used to instruct the compiler to stop compiling a Verilog description that begins with a /* synthesis

translate_off */ statement and ends with a /* synthesis translate_on */ statement.

Syntax Specification

Verilog	`/* synthesis translate_off */`
	`/* synthesis translate_on */`
	or
	`// synthesis translate off`
	`// synthesis translate on`

Verilog Example

```
module test(
      input [5:0] a,b,
      output [6:0] dout,
      output [11:0] Nout);
      assign dout = a + b;
      /* synthesis translate_off */
      assign Nout = a * b;
      /* synthesis translate_on */
endmodule
```

If this property is not used in the example, the multiplier logic will be inferred, as follows

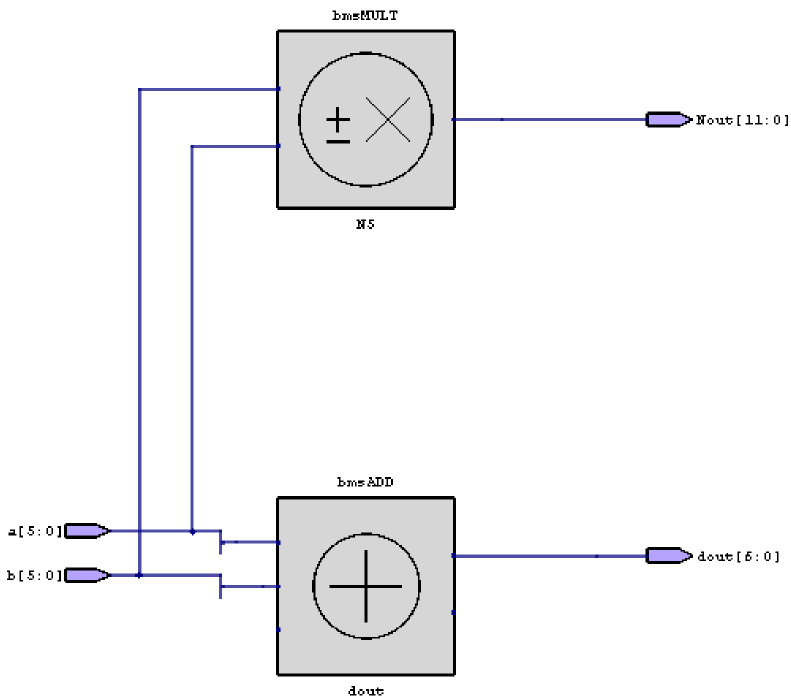


Figure 5-44 The translate_off/on property is not applied

With the translate_off/translate_on attributes applied, the multiplier logic is not inferred and the

Nout signal is left hanging, as shown in the figure below.

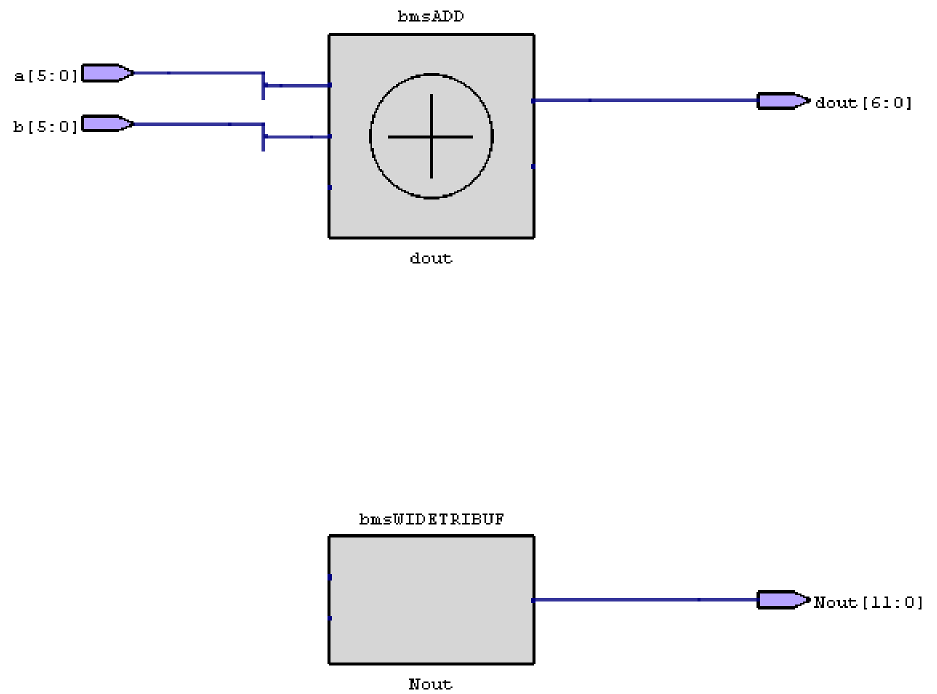


Figure 5-45 Applying the translate_off/on Property

6. SWITCHING FROM SYNPLIFY PRO TO ADS

6.1 Selecting the ADS Synthesis Tool

When creating a new PDS project, in the "New Project Wizard", when selecting the "Device" model, select "Synthesis Tool: " drop-down menu, select "ADS.

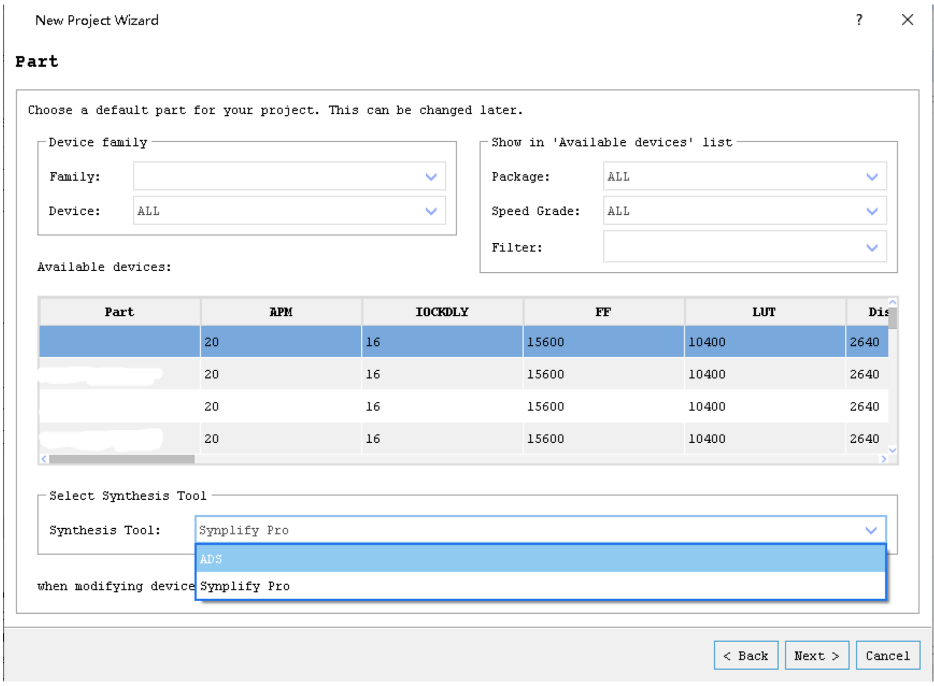


Figure 6-1 Integrated tool selection

To switch an existing PDS project to "ADS": Right-click on the device name under "Navigator" and select "Project Settings", or "Right-click" on any action under "Flow" and select " Project Settings", then select the "Part" page on the left, and in the "Synthesis Tool:" drop-down menu below the Part page, select "ADS".

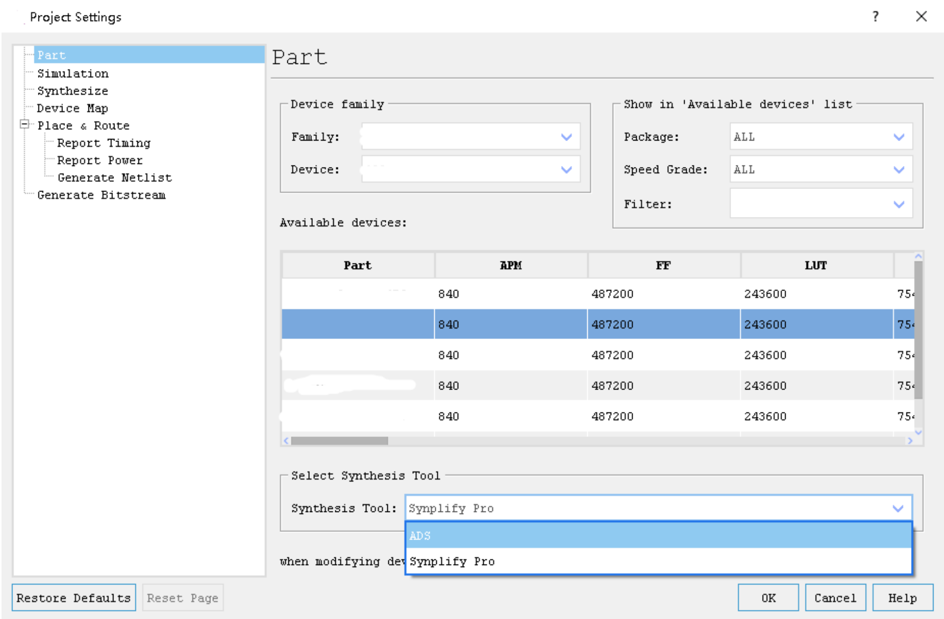


Figure 6-2 Integrated Tool Selection in Project Setup

6.2 Synthesis Flow Differences

Synplify Pro, a third-party synthesis tool, has only one subflow in the PDS flow, while ADS Synthesis has two subflows in the main PDS flow, Compile and Synthesize.

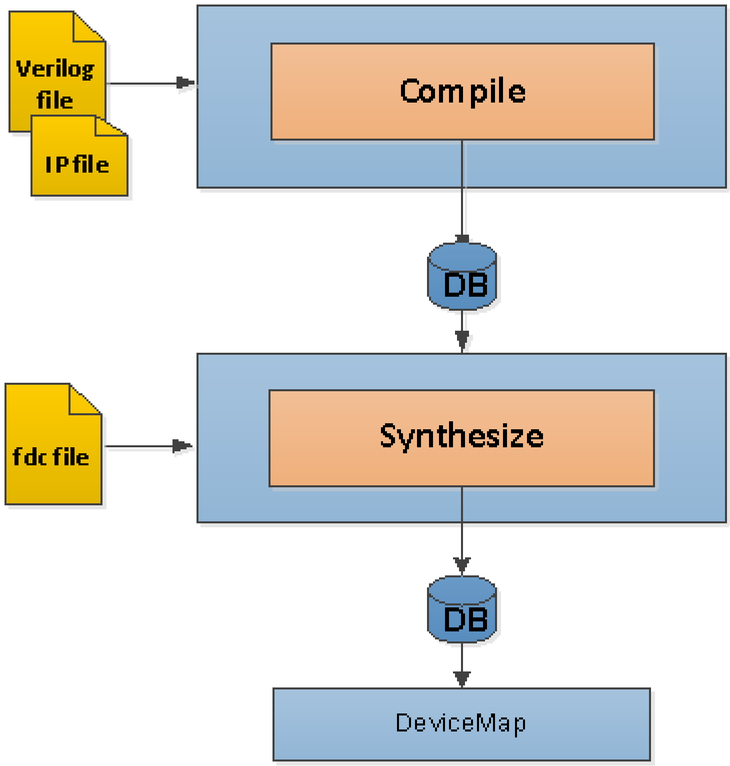


Figure 6-3 ADS synthesis process

ADS synthesis is divided into two sub-processes, Compile and Synthesize, to facilitate user awareness. fdc constraint files are read in during the Synthesize phase and act on the RTL netlist generated after Compile.

Note: The synthesize fdc (FPGA Design Constraint) constraint file is applied to the RTL netlist after Compile, regardless of whether it is Synplify Pro or ADS.

6.3 HDL Language Differences Supported

Synplify Pro for Pango is a mature FPGA synthesis tool developed by Synopsys, which supports the synthesizable subsets of Verilog-2001, SystemVerilog, VHDL-1993, some VHDL-2008 and VHDL-2019 languages.

ADS is a self-developed, in-development FPGA synthesis tool from Violet. It currently supports a synthesizable subset of Verilog-2001 and a limited SystemVerilog syntax.

Current supported SystemVerilog syntax: **input/output port declarations in the form of two-dimensional arrays, $clog2, $signed, $unsigned system functions**.

In PDS_2019.4 version, the default language supported by ADS is Verilog-2001. When using a project with SystemVerilog syntax, you need to manually turn on **SystemVerilog** support.

"Right-click" Compile, select "Project Settings", and check SystemVerilog in it.

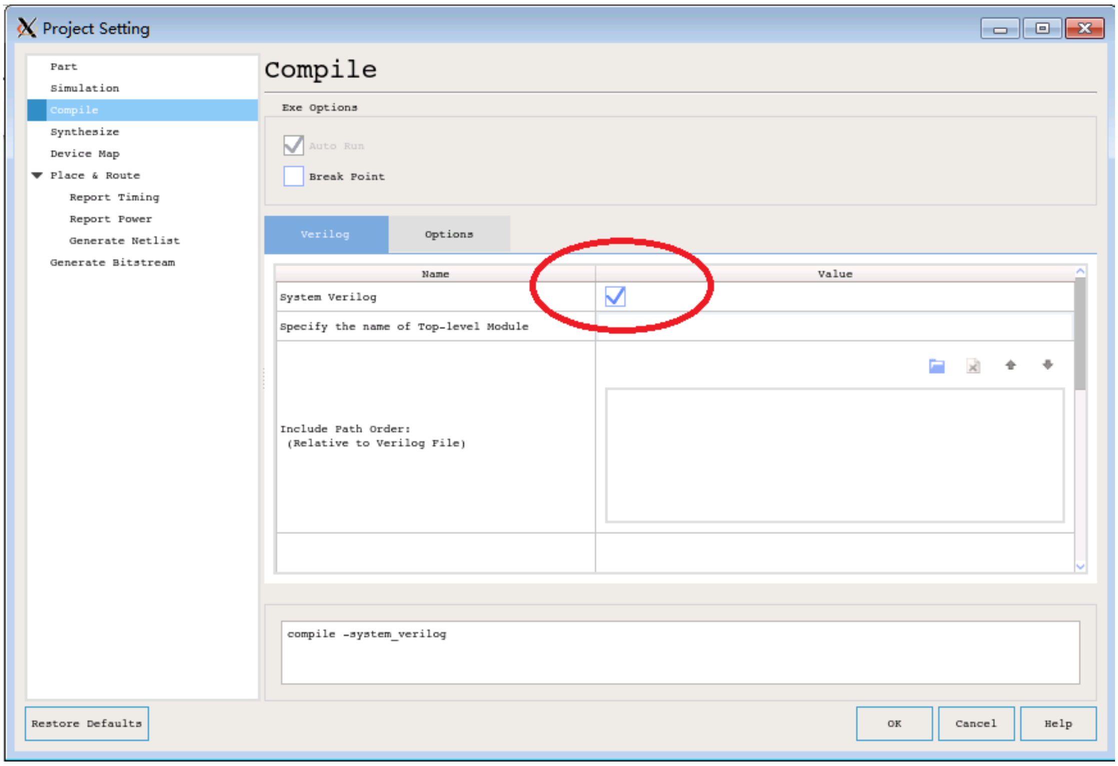


Figure 6-4 SystemVerilog Support

6.4 Possible FDC Constraint File Modifications

As described in Section 2, Synthesis Flow Differences, the FDC constraint file is applied to the post-compile RTL netlist. While Synplify Pro and ADS are two independently developed synthesis tools, the generated post-compile RTL netlist may not contain the design object (e.g., instance name) constrained by the FDC constraint file. In this case, you must manually modify the FDC constraint file to match the ADS synthesis tool.

ADS provides an FDC Constraint Check tool, right-click on "FDC Constraint File" and select "Constraint Check" from the pop-up menu to report the unsuccessful FDC commands and generate a Constraint Check Report (.ccr).

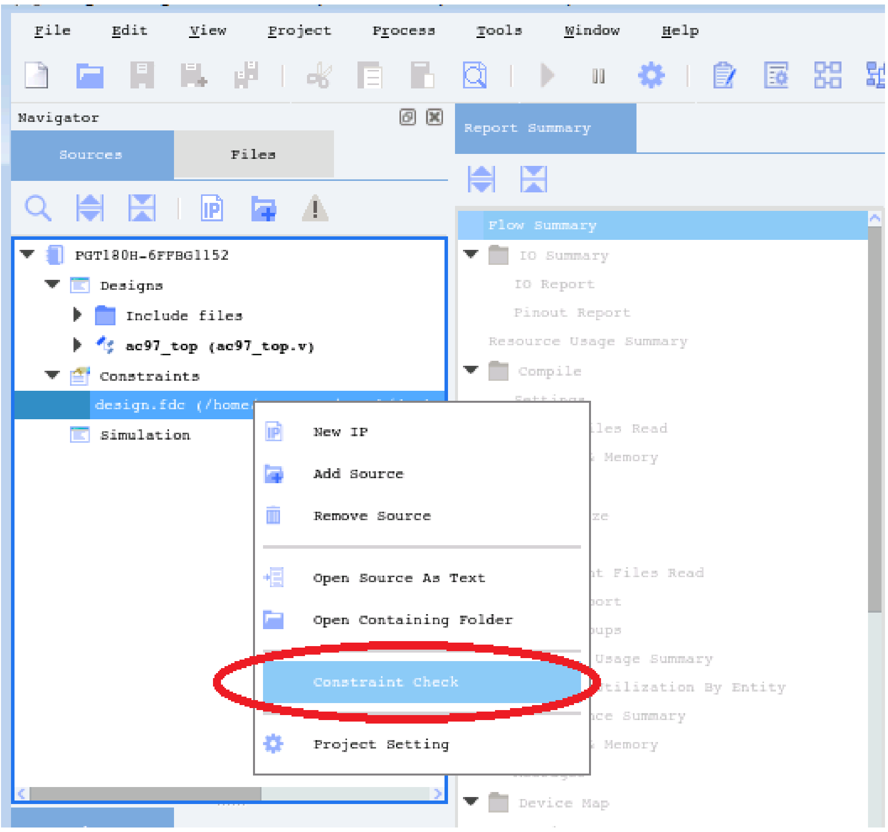


Figure 6-5 Constraint Check Constraint Check

You can use the RTL Schematic tool to locate the name of the design object to be constrained.

One is to select "View RTL Schematic" from the "Schematic View" submenu of the "Tools" menu, as shown in the figure below.

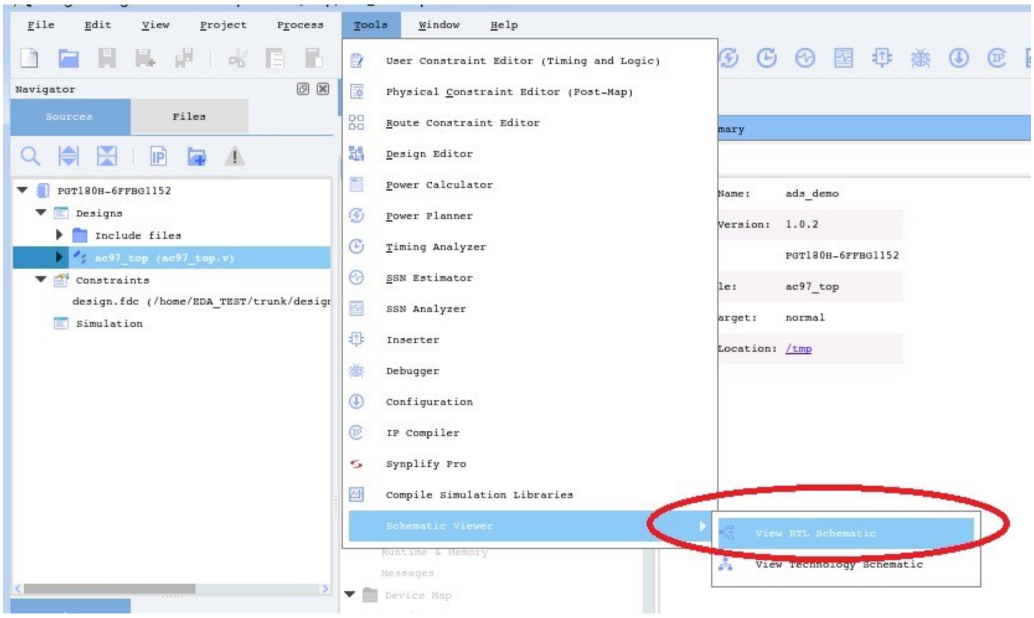


Figure 6-6 RTL Schematic Tool

Second, in the toolbar icon, click the icon in the red box below.

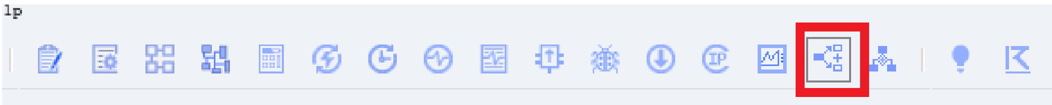


Figure 6-7 Selecting RTL Schematic in the toolbar

After opening the "RTL Schematic", you can use the "Design Browser" and the search function to find the "design object" that needs to be constrained. design object" to be constrained.

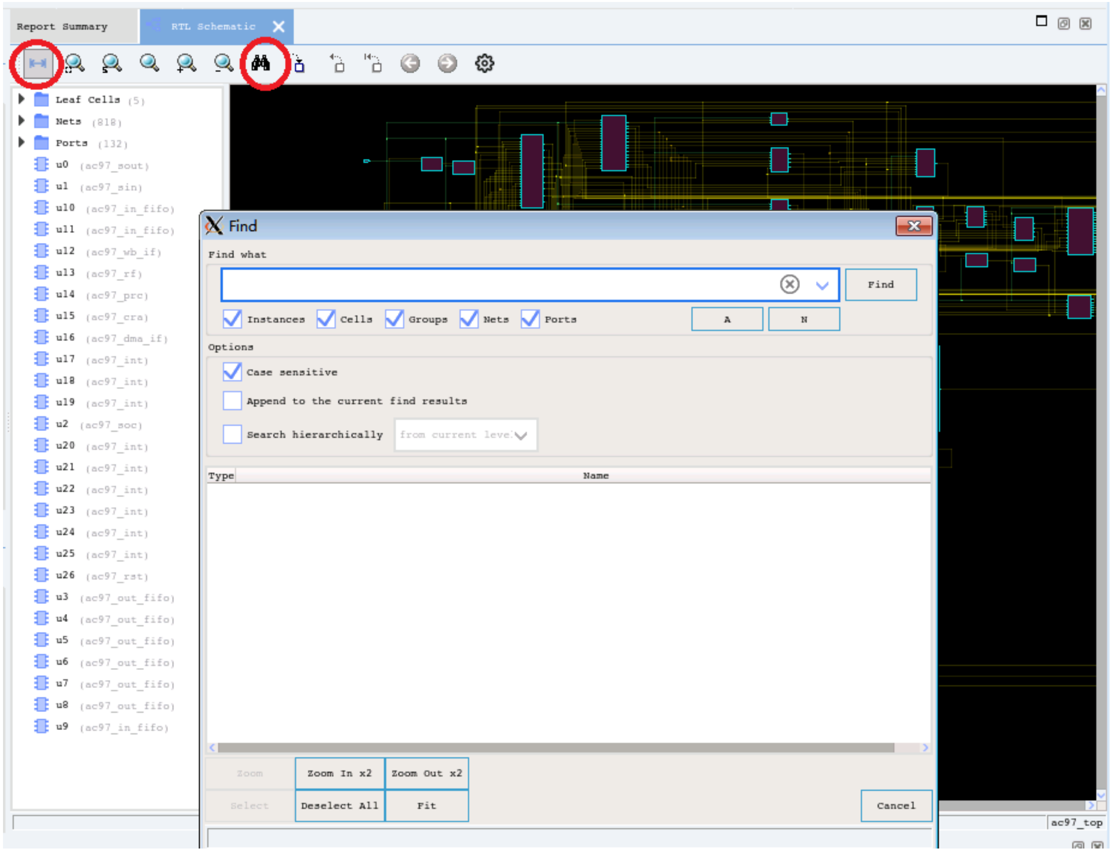


Figure 6-8 Design Browser Functions

6.5 Warnings and Cautions

6.5.1 Error E: Array port <xx> is not supported. Enable SystemVerilog features first.

The two-dimensional array port declaration is SystemVerilog syntax and requires that SystemVerilog be enabled first.

Explanation: Synplify Pro may have been developed with SystemVerilog and confused the boundary between Verilog-2001 and SystemVerilog syntax, while ADS strictly distinguishes by standard.

6.5.2 Error E: Verilog-4088: Index <xx> of <xx> is out of range

ADS strictly checks the bit-select, part-select of vectors and reports an error if the bit-select or part-select of a constant exceeds the range of the variable definition.

Synplify Pro does not strictly check for "out of range" errors, which are reported in some cases and not in others.

Synplify Pro for: CS101:@E: Index <xx> is out of range for variable <xx>.

6.5.3 Error E: DRC-0006: Pin <pin-name> of inst <inst-name> is floating

The ADS is designed to check whether each input pin is connected to a net and whether each net has a driver and will report this error if the rules are not met.

Syn_unconnected_inputs should be displayed for pins explicitly required to be floating.

6.5.4 Unnamed Generate Block Name

According to the Verilog-2001 standard, if the generate block does not display a specified name, the design object is automatically named genblk<n> (where n is the number of occurrences in the same level of the generate block) to access the design object in the generate block in the order in which it appears in the generate block. generate)

The following is taken from the IEEE 1364-2005 Verilog HDL Reference Manual

12.4.3 External names for unnamed generate blocks

Although an unnamed generate block has no name that can be used in a hierarchical name, it needs to have a name by which external interfaces can refer to it.

Each generate construct in a given scope is assigned a number. The number will be 1 for the construct that appears textually first in that scope and will increase by 1 for each subsequent generate construct in that scope. All unnamed generate blocks will be given the name "genblk<n>"

Figure 6-9 Unnamed generate block naming standard

7. CONTACTS

Контакты для технических и коммерческих вопросов
ООО «Инэк»
г. Санкт-Петербург, ул. Яблочкова, д. 20, литер А, оф. 504
contact@innek.ru
+7 (812) 200-40-37